Universidad Carlos III de Madrid
Grado en Ingeniería Telemática

TRABAJO FIN DE GRADO

Desarrollo de aplicaciones Android Wear

Autor: Germán Schiavón Matteo
Tutora: Celeste Campo Vázquez

Resumen

La tecnología Android Wear ha cambiado la forma en la que vemos los wearables, como las gafas, los relojes, las pulseras, etcétera. Son complementos de vestir que han estado con nosotros durante mucho tiempo, pero es ahora cuando Google les ha dado una funcionalidad totalmente desproporcionada en comparación con su pequeño tamaño, como en el caso de las Google Glass. Todo esto ha sido posible gracias a la reducción considerable que han tenido los diferentes componentes hardware, como por ejemplo la batería.

A partir de la versión 5, Android ha abierto su abanico de posibilidades, introduciéndose en el mercado de las televisiones con Android TV, en el de los ordenadores de a bordo de los coches con Android Auto, y finalmente en el mercado de los wearables con Android Wear.

En esta memoria de trabajo de fin de grado, trataremos exclusivamente sobre Android Wear, y más concretamente, sobre los relojes inteligentes. Intentaremos dar una visión general sobre esta tecnología. Se hablará de sus características principales tanto de software como de hardware. Se hará especial énfasis en el mundo de las notificaciones y en el paso de mensajes entre el reloj y el teléfono, explicando cómo es posible la comunicación entre ambos. Se realizará una comparativa de mercado con uno de sus principales competidores, el Apple Watch.

Este trabajo incluye el desarrollo de dos aplicaciones Android Wear creadas expresamente para explicar y entender la comunicación entre el reloj y el teléfono y para ilustrar el alcance de esta novedosa tecnología.

Abstract

Nowadays Android Wear technology has changed the way we see wearables such as glasses, watches, wrist bands and so on. Wearables have been among us for a very long time, but it is only now with the awesome features that Google has added to them that we realize that even though they are so small, they can be very useful and powerful. Take for example Google glasses.

All of the advancements in these devices are only possible thanks to an extreme reduction of technology, such as the development of more efficient batteries.

Ever since its 5th version, Android has opened its range of possibilities getting into different industries like for example television with Android TV, car computers with Android Auto, and the wearable market with Android Wear.

In this report, we will talk about Android Wear, specifically about smartwatches. We will try to give an overview about this new technology and we'll describe their main features, both in software and hardware. We'll talk about notifications, and how data is synchronized and sent from the smartwatch to the phone. All of these are important features that we will explain in detail. Also, we will compare Android Wear to the Apple Watch, its most important competitor in the market.

This project includes the development of two Android Wear applications created to explain and understand the communication between the watch and the phone and also to show the powerful Android Wear could be.

Índice general

Índice de figuras

Índice de tablas

Índice de bloques de código fuente

1. Introducción

En este apartado se presentará un visión general del proyecto, explicando la motivación que llevó a su desarrollo, los objetivos que se desean alcanzar, y las distintas fases de desarrollo por las que atravesó.

1.1 Visión General

En esta memoria se intentará dar una visión global sobre todo lo que rodea a Android Wear, haciendo especial hincapié en los relojes inteligentes. Recordamos que la tecnología Android Wear puede ser usada en otros dispositivos distintos a los relojes.

Se recorrerá la historia de los wearables, empezando con el que mayor revuelo causó, las Google Glass, y terminando con los relojes inteligentes. Estos pequeños dispositivos han cambiado nuestra vida en muchos aspectos, han dejado de ser un complemento de moda para convertirse en aparatos con una gran funcionalidad capaces de registrar a través de sensores nuestra presión sanguínea, nuestra actividad física realizada, y muchas más.

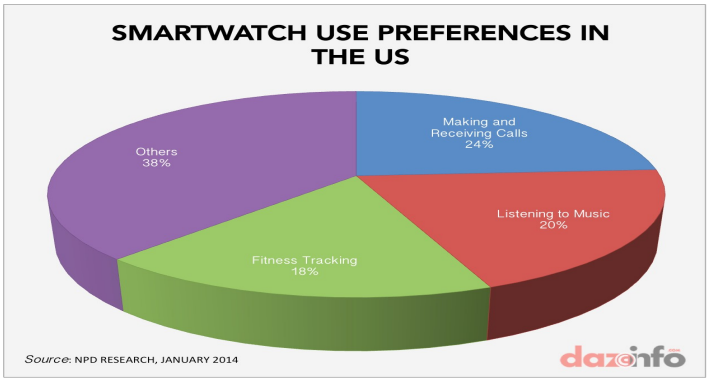

Figura 1. Preferencias de uso de los relojes inteligentes [1].

Para explicar algunas de sus características se han implementado dos aplicaciones que utilizan dos de las grandes funcionalidades de las que dispone Android Wear:

- El envío y sincronización de información
- Acciones mediante el uso de la voz

Ambas aplicaciones hacen uso del envío y sincronización de información, esto es de vital importancia, pues la comunicación entre el reloj y el teléfono es lo que hace que todo esto sea posible. Si no hay comunicación entre ambos, el wearable pierde el cien por cien de su funcionalidad. Es por esto, por lo que se han desarrollado aplicaciones haciendo uso de la comunicación reloj-teléfono.

El uso de acciones mediante el uso de la voz es otro de los aspectos más importantes cuando estamos desarrollando aplicaciones para dispositivos wearables, debido a que éstos no disponen de teclado.

1.2 Motivación

La motivación principal que ha llevado al desarrollo de este proyecto ha sido principalmente utilizar una tecnología nueva como Android Wear, anunciada por primera vez el 18 de marzo de 2014, tan nueva que a medida que se desarrollaba el proyecto iban surgiendo cambios importantes[46].

El hecho de trabajar con una tecnología con menos de dos años supone un reto importante, debido a la falta de información, información confusa, menos ejemplos en los que fijarse, etcétera.

También el desarrollo de este proyecto coincidió con la liberación del nuevo entorno de desarrollo de Android Studio, lo que supuso un gran número de horas hasta conseguir manejarlo correctamente.

Otro de los aspectos más motivadores ha sido el hecho de trabajar con una tecnología que parece sacada de un cómic de ciencia ficción. El hecho de hablarle a un reloj y que este ejecute una acción en el teléfono parecía impensable hace unos años. La comunicación entre el reloj y el teléfono es un tema interesante que tendrá un destacable impacto en los próximos años.

1.3 Objetivos

El objetivo principal de este proyecto es acercar de una manera teórico-practica la tecnología Android Wear. Para ello se han desarrollado dos aplicaciones, que tratan sobre los temas más importantes en Android Wear.

El objetivo teórico que se pretende alcanzar en esta memoria, es explicar qué es Android Wear, de dónde viene y a dónde va. Esto se traduce en cómo está actualmente Android Wear integrado en la sociedad, la aceptación que tiene y el uso que se puede hacer de esta tecnología. Esto es importante debido a que poseen una gran funcionalidad que puede ser aplicada a distintos ámbitos. El otro aspecto importante es saber si esto se va a quedar aquí o por el contrario van a seguir evolucionando hasta convertir Android Wear en un objeto indispensable.

Los objetivos más generales son:

1. Estudiar la tecnología Android Wear centrándonos en las comunicaciones entre el dispositivo wearable y un dispositivo móvil.

2. Conocer en detalle el entorno de desarrollo de esta tecnología, que prácticamente pasó de la versión Beta a la versión definitiva durante las primeras fases del proyecto[47].

3. Comparar esta tecnología respecto a otras existentes en el mercado.

4. Diseñar, desarrollar y probar aplicaciones Android Wear que hagan uso de la comunicación entre este dispositivo y el móvil, y también que exploten la comunicación con voz.

5. Realizar un estudio sobre en marco socio-económico y el marco regulador actual de los wearables.

6. Documentar todo el trabajo realizado.

1.4 Etapas de desarrollo

El proyecto ha pasado por etapas de desarrollo muy diferenciadas.

Etapa 1.

La fase inicial corresponde con la búsqueda y recopilación de la información. Dominar una tecnología siempre lleva tiempo, y si añadimos que Android Wear en el comienzo de este proyecto tenía menos de un año, todavía más.

Etapa 2.

Una vez recopilada la información, la siguiente fase es entender el nuevo entorno de desarrollo Android Studio, liberado en el mismo momento del inicio del proyecto. Realizar aplicaciones sencillas de prueba, utilizar el emulador. Conseguir conectar el teléfono con el emulador fue uno de los puntos claves.

Etapa 3.

Después de los primeros pasos con Android Studio, y conocer las características de Android Wear, ya estaba en disposición de pensar en aplicaciones que exploten las principales características de Android Wear, como el paso de mensajes entre el reloj y el teléfono y la comunicación mediante la voz con el reloj. Se definen las líneas generales de la aplicación Find Your Phone. Ya se dispone de un dispositivo Android Wear real, por lo que no se vuelve a usar el emulador.

Etapa 4.

Después de diseñar la aplicación Find Your Phone, comienza su desarrollo con su posterior fase de pruebas. Una vez completadas estas fases se da por terminada.

Etapa 5.

Tras la primera aplicación, se piensa en una segunda que complemente la primera, en el sentido de utilizar otra de las novedosas características de Android Wear, el uso de la voz. Se desarrolla la aplicación Social Media Opener.

Etapa 6.

Con la fase de desarrollo completa, comienzo con la redacción de la memoria. Fase en la que se utiliza parte de la información recopilada en la primera etapa más información nueva.

Etapa 7.

Estudio del marco socio-económico y marco regulador de los dispositivos wearables.

1.6 Contenidos

En este apartado se explicarán mediante una breve descripción el resto de los capítulos que forman esta memoria.

1. Introducción.

En este capítulo se expone una visión general del proyecto, la tecnología, los objetivos y las motivaciones que han hecho que se llevara a cabo.

2. Estado del arte.

En este capítulo se hablará de la historia, la actualidad, y las tendencias en los próximos años de los wearables. Se realiza una comparativa con la competencia más directa, el Apple Watch.

3. Android Wear.

En este capítulo se hará una introducción a Android Wear, comentando sus principales características, cómo diseñar aplicaciones para wearables y requisitos para ello. También incluye un repaso sobre la plataforma Android, destacando sus puntos característicos y desmenuzando la arquitectura que la conforma.

4. Configuración del entorno y primeros pasos.

En este capítulo se encuentra un pequeño tutorial para comenzar a utilizar el entorno de desarrollo, creación de un primer proyecto y uso del emulador

5. Aplicación Find Your Phone.

En este capítulo se explica en detalle de los diferentes componentes que forman parte de la aplicación. Se detallan sus características principales y su correcto funcionamiento.

6. Aplicación Social Media Opener.

En este capítulo se explica detalle de los diferentes componentes que forman parte de la aplicación. Se detallan sus características principales y su correcto funcionamiento.

7. Planificación y desarrollo.

En este capítulo se incluye un diagrama de Gant con las distintas etapas por las que ha pasado el proyecto, explicadas con detalle. También se incluye el presupuesto del proyecto.

8. Entorno socio-económico.

En este capítulo se realiza un estudio sobre el marco socio-económico en el que se encuentran los wearables actualmente.

9. Marco regulador

En este capítulo se encuentra un análisis sobre el marco regulador al que son sometidos los dispositivos wearables.

10. Conclusiones.

En este capítulo se hace una valoración personal del proyecto, comentando los puntos fuertes y débiles de la tecnología Android Wear. Posibles mejoras al proyecto.

11. Referencias.

En este capítulo estarán todas las fuentes que han sido consultadas para la realización del proyecto.

12. Glosario.

En este capítulo se encontrarán todas las siglas que han sido utilizadas en este documento.

2. Estado del arte

En este apartado se va a hacer un pequeño repaso a la historia de los wearables, la actualidad de los dispositivos tanto móviles como wearables, pues su destino esta fuertemente ligado, y también se hará un estudio de las aplicaciones que hay en el mercado con el fin de crear algo que despierte el interés en los usuarios.

2.1 Dispositivos móviles

Un dispositivo móvil es un pequeño dispositivo de computación, normalmente lo suficientemente pequeño para que pueda caber en la mano, con pantalla y un teclado, ya sea táctil, o no, y que pese menos de 900 gramos.

Una vez definido que es un dispositivo móvil, hablemos del motor de éstos, los sistemas operativos. Android es actualmente el sistema operativo más usado, con una gran comunidad de desarrolladores creando aplicaciones.

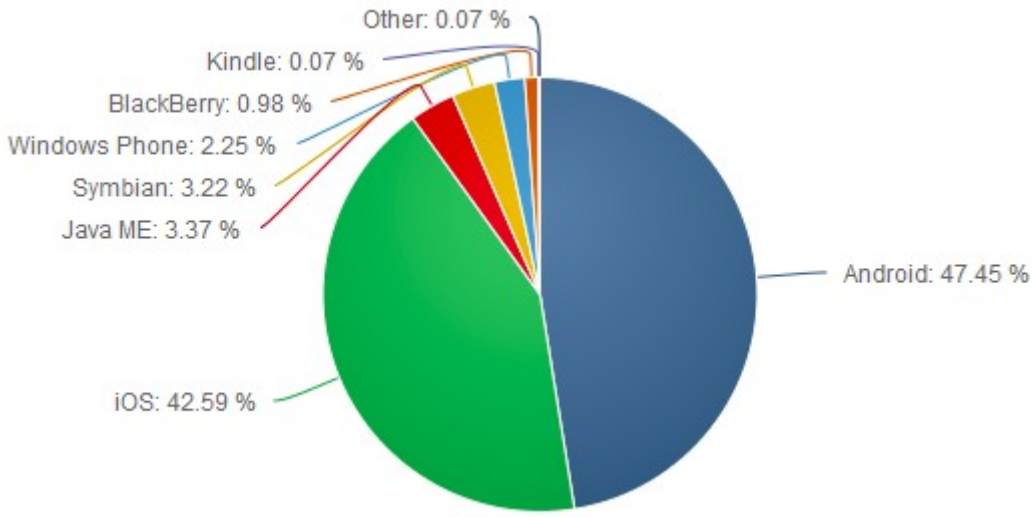

Figura 2. Sistemas operativos más usados[2].

Como se puede ver en este gráfico, la mayoría de los dispositivos móviles funcionan con el sistema operativo de Android, por lo tanto, desarrollar aplicaciones para esta plataforma es sinónimo de tener la posibilidad de alcanzar mayor cuota de mercado.

En el caso de este proyecto, no se ha tenido que elegir ninguna plataforma en concreto, como suele pasar cuando se desarrollan aplicaciones para teléfonos móviles. Android Wear ya pertenece a la plataforma de Android, y en el momento de su lanzamiento en marzo de 2014, la competencia la constituían los relojes inteligentes de Samsung, y de Pebble básicamente. En 2015 se suma un nuevo gran competidor, el Apple Watch de la compañía Apple.

2.2 Historia de los wearables

La historia de los wearables, aunque nos pueda parecer extraño, viene desde muy atrás, con los primeros relojes de mano. Después llegaron los demás, gafas, pulseras y cualquier complemento de moda que se nos pueda ocurrir. Pero, ahora si, mucho dista aquel reloj de mano con los nuevos relojes inteligentes de los que hablaremos más adelante.

El nuevo concepto de wearable dio el salto al mundo con una demostración épica en la que el gigante Google presentó las Google Glass el 27 de junio de 2012 [4]. Unas gafas, un elemento hasta ahora solo usado para proteger los ojos del sol y para mejorar la visión de las personas, se convertían en un punto de convergencia entre la realidad aumentada y la visión computacional.

Después del revuelo que causó esta demostración, Android no iba a dejar pasar esta oportunidad y en 2014 presenta la tecnología Android Wear. Ésta se utilizará en los relojes inteligentes cambiando el concepto que teníamos de los relojes hasta el momento. Aunque al principio muchos pensaban que seria un elemento sustitutivo de los teléfonos, éste sólo sería una extensión del teléfono.

Figura 3. Evolución de los relojes, desde el reloj solar hasta el reloj inteligente o smartwatch

Un reloj inteligente o smartwatch, no es más que un reloj de pulsera con una funcionalidad mejorada que va más allá de mostrar la hora. Como se puede ver en la Figura 3, mientras las primeras funcionalidades extras eran funciones básicas, como la calculadora, traductores, ahora un reloj inteligente es algo más parecido a un ordenador [3].

Un smartwatch puede ejecutar aplicaciones, también incluyen distintos sensores como acelerómetro, giroscopio, gps, altímetro, barómetro, medidor de pulsaciones, etcétera.

2.3 Fabricantes de hardware para Android Wear

Uno de los factores a destacar, es que Android Wear consta con un gran número de grandes empresas que se dedican a fabricar dispositivos para esta tecnología. Algunos ejemplos de estas empresas son:

- LG
- Sony
- Motorola

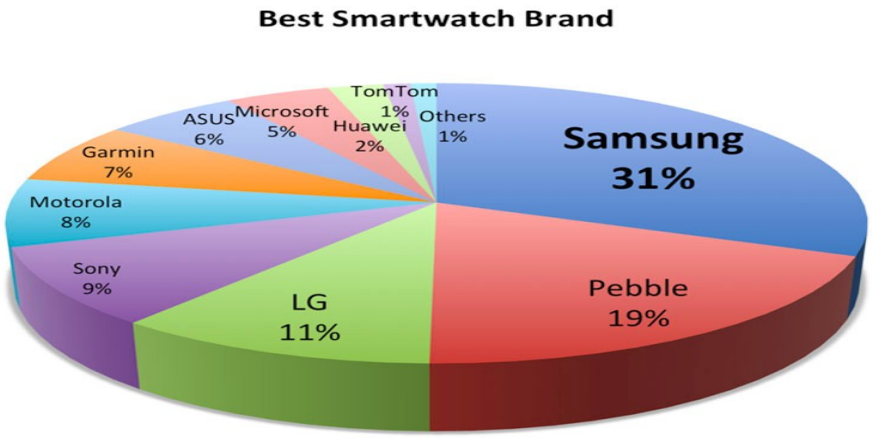

Figura 4. Mejor marca de smartwatches en 2014[6].

En la Figura 4 se puede ver que Samsung está catalogada como la mejor marca de relojes inteligentes según el *smartwatch group[5]*. A parte de estar considerada la mejor marca, también es la que más vende. Suelen ser conceptos que van unidos, aunque no en todos los casos. Sus ventas son de alrededor de unos 1,2 millones de unidades vendidas en 2014[7]. Ésta empresa ya se ha lanzado a por la segunda generación de relojes inteligentes con su nuevo producto Gear S y su pantalla curva [8]. El sistema operativo de los relojes Samsung es Tizen[9], es un sencillo sistema que esta destinado a servir como complemento a los teléfonos de la gama Galaxy de Samsung. La segunda compañía es Pebble, con 700.000 unidades vendidas, marca que realiza su propio smartwatch, tanto hardware como software y no tienen nada que ver con Android Wear.

El mercado está a la espera de los resultados definitivos del Apple Watch en 2015, que ha entrado muy fuerte en el mercado, con un producto consistente, y que seguro que va a causar un terremoto entre las compañías. Los analistas del mercado más optimistas prevén vender más de 20 millones de unidades antes de que acabe el 2015.Aunque Apple no da detalles del primer semestre del año, parece que las ventas no transcurren como esperaban los analistas [10].

2.4 Comparativa Android Wear con Apple Watch

En este apartado vamos a comparar a los dos grandes rivales del momento en cuanto tecnología de relojes inteligentes se refiere.

Cabe destacar que el Smartwatch de Apple es bastante más reciente, concretamente el 26 de junio de 2015 y ha podido utilizar en su beneficio la experiencia de sus competidores en el mercado.

Antes de entrar más en detalle, una de las principales diferencias es que Apple desarrolla tanto el dispositivo físico como su funcionalidad interna (Hardware y Software).

Los temas que vamos a tratar son los siguientes:

- Diseño externo
- Diseño de la interfaz gráfica y usabilidad
- Aplicaciones
- Batería
- Notificaciones

1. Diseño externo.

- El Apple Watch por ahora consta de un modelo único, con distintas correas y colores [11], todavía no se conoce si habrá más variedad en los diseños. Tiene una forma rectangular con los bordes redondeados lo que le da una forma elegante.

Figura 5. Diseño Apple Watch[12].

- En cuanto al diseño de los relojes con Android Wear, encontramos una variedad mucho más amplia. Éste corre a cargo de numerosos fabricantes como LG, Motorola, y Sony entre otros. El diseño más reclamado por los usuarios es el del Moto360, debido a su formato circular y a su gran elegancia.

Figura 6. Diseño del reloj Moto360[13].

Al haber más variedad y más fabricantes implicados en la creación de dispositivos, Android Wear tiene ventaja en este aspecto.

2. Interfaz gráfica y usabilidad.

- El Apple Watch tiene una pantalla principal similar al iPhone. Esto es, un menú que consta de iconos circulares, como en la Figura 6, de todas las aplicaciones de nuestro reloj. Moviendo la corona se pueden hacer más grandes o más pequeños. También utiliza mucho los fondos negros lo cual hace que de la sensación de que se aprovecha toda la pantalla.

- Android Wear, utiliza un sistema muy similar al de las tarjetas de Google Now[14]. Google Now es un sistema que muestra al usuario la información más relevante dependiendo de las búsquedas que realizan, y las enseña por pantalla en modo de tarjetas[15] . Mediante el desplazamiento lateral de nuestro dedo accedemos a las distintas aplicaciones. Este desplazamiento también sirve para cerrar las mismas.

3. Aplicaciones.

- En cuanto a aplicaciones, Android Wear tiene una ligera ventaja debido a que lleva más tiempo en el mercado, pero las aplicaciones más utilizadas ya están disponibles para ambas plataformas.

4. Batería.

- Ambos están bastante igualados en este aspecto. Con un uso normal, deberían durar unas 12 horas aproximadamente lo que hace que sea obligatorio cargarlo todas las noches.

5. Notificaciones.

- Las notificaciones es uno de los aspectos más importante, Whatsapp y Facebook fueron de las primeras grandes aplicaciones en dar soporte a Android Wear, esto es, en mostrar las notificaciones en el reloj de manera automática.

- Tanto en Android Wear como con Apple, las notificaciones pueden ser abiertas de varias formas y pueden contener acciones:

    - Leerlas desde el reloj
    - Indicar que quieres abrir la notificación en el teléfono, desde el reloj.
    - Responderla mediante voz, en formato texto, no una nota de voz.

En cuanto al aspecto económico, no hay que olvidarse que el coste oficial de los Apple Watch a día 9 de marzo de 2015 varía entre 549 y 1099 dólares. Apple tiene un modelo más económico llamado Sport, 349 dólares[16], que será el encargado de competir con los Android Wear debido a que se encuentran en un precio similar.

2.5 Tendencias

El futuro para esta tecnología es muy esperanzador. Las compañías encargadas de crear hardware ya están trabajando en las nuevas generaciones de wearables, añadiendo nuevas funcionalidades.

Hay artículos[6], que declaran que esta tecnología se ha adelantando 5 años a nuestro tiempo. Debido a que próximamente con nuestros relojes, vamos a poder hacer la compra, abrir puertas, y muchas cosas más. A pesar de que los relojes inteligentes ya disponen de la funcionalidad para hacer nuestra vida más fácil, falta mucho para que esta funcionalidad sea aprovechada al cien por cien. Es tecnológicamente posible hacer puertas, o integrar sistemas en las tiendas para que los relojes se puedan comunicar, pero por ahora, económicamente hablando, el coste es muy alto.

Pero pese a estas críticas negativas se prevé que los smartwatches inunden el mercado en los próximos años.

Se espera que para finales del 2015 más del 50 por ciento de los usuarios de pulseras con fines deportivos hagan el cambio a los relojes inteligentes. Los relojes con características capaces de monitorizar la actividad física serán los que mejor aceptación tengan.

Los fines de los wearables pueden llegar a ser muy variados[17], como los siguientes:

- Gafas con más visión
- El casco con un sistema para hacer crecer el pelo
- La máscara de belleza
- El entrenador mental
- y muchos más

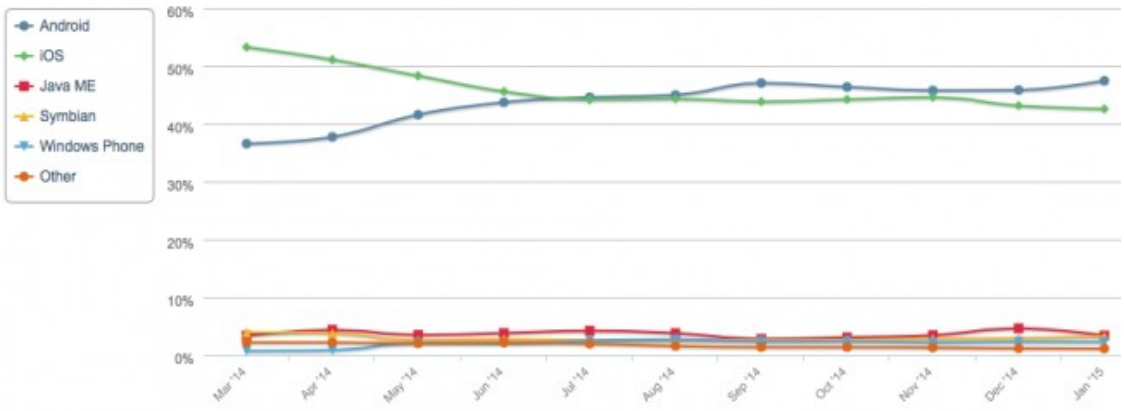

Figura 7. Tendencia de los sistemas operativos para móvil[2].

En la Figura 7, podemos ver la tendencia de los sistemas operativos para móvil para el año actual, 2015. Esta gráfica es importante, y no hay que olvidarse de que el teléfono y el reloj van de la mano. Una caída de Android en el sector de la telefonía móvil supondría una caída en los relojes con sistema operativo Android Wear[2].

3. Android Wear

En este punto se hará una introducción a la tecnología Android Wear, hablando desde sus inicios hasta la actualidad, se mostrarán las principales características y los primeros pasos desarrollando aplicaciones Android Wear. Se incluye una introducción a la plataforma Android.

3.1 Introducción Android

Android es un sistema operativo basado en linux. Surgió principalmente para dar soporte a las cámaras fotográficas profesionales, aunque rápidamente se extendió a los teléfonos móviles. Posteriormente se expandió a otros sectores como tablets, relojes, automóviles y televisores entre otros.

Android es un proyecto Open Source, de hecho, la forma correcta de referirse a este sistema operativo es Android Open Source Project. Esto se traduce en que es un código libre en el que cada uno lo puede usar como quiera. En Android se programa de forma nativa en C/C+, pero un alto porcentaje de aplicaciones están en Java, también se pueden utilizar otros lenguajes como C#, visual, etcétera.

3.1.1 Historia

En 2003 un grupo formado por cuatro jóvenes de la localidad de Palo Alto, Andy Rubin, Rich Miner, Nick Sears y Chris White, con una visión diferente de los dispositivos móviles, deciden crear Android Inc, que será la primera amenaza competidora en sistemas operativos móviles para Windows Phone y Symbian.

La realidad, como hemos mencionado anteriormente, el sistema operativo estaba exclusivamente orientado a cámaras fotográficas, pero al darse cuenta de que el mercado de los sistemas operativos para móviles tenía poca competencia, decidieron introducirse en él. En 2005 Android Inc es comprada por Google y el primer móvil que incluyó esta tecnología fue el HTC Dream en octubre de 2008.

La historia de Android comienza con la liberación de Android beta en noviembre de 2007. La primera versión comercial, Android 1.9, fue liberada en septiembre de 2008. Un hecho importante fue a finales de 2007 cuando la Open Handset alliance, grupo formado por distintas compañías como Google, Sony, HTC y Samsung entre otras, anuncian que están trabajando en un sistema operativo común y abierto para una plataforma móvil.

El HTC Dream, que fue el primer smartphone con sistema operativo Android, vendió más de 1 millón de terminales en Estados Unidos. A partir de entonces las ventas de dispositivos móviles con sistema operativo Android se disparan y superan a IOS y Windows Phone.

3.1.2 Historial de actualizaciones

La historia de versiones del sistema operativo Android comienza con la liberación de la versiones conocidas como alpha y beta en noviembre de 2007. Hay que destacar que en ese momento no había dispositivos comerciales que las pudieran utilizar.

Desde el 22 de octubre del 2008[9], las versiones de Android tienen nombres de cosas dulces, su teoría es que hacen que nuestra vida sea más dulce y sus nombres siguen un orden alfabético: Apple Pie, Banana Bread, Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat y Lollipop. La actualización más reciente es Android Marshmallow, que fue anunciado oficialmente en mayo de 2015, durante la conferencia I/O de Google[18].

3.1.3 Arquitectura

La arquitectura de Android contiene una pila de software que contienen distintas capas cuyo principal objetivo es abstraer en todo momento el hardware, facilitando así el desarrollo de aplicaciones[19][20].

Figura 8. Arquitectura de Android

La ventaja de esta arquitectura es que permiten acceder a las capas inferiores usando librerías específicas.

Aplicaciones

En esta capa se encuentran todas las aplicaciones nativas de Android más las aplicaciones de terceros que el usuario instale. Las aplicaciones base incluyen alarma, buscadores, calculadora, calendario, cámara, reproductor media, cliente de correo electrónico, etcétera. Todas estas aplicaciones tienen acceso a la API, los servicios y las librerías de las capas anteriores.

Entorno de trabajo de Android:

En este nivel están alojadas las APIs que cualquier aplicación desarrollada en Android, ya sea nativa del sistema, creada por terceros o por el mismo usuario, tendrá que utilizar.
De este modo se unifica el entorno de trabajo, con la misión principal de facilitar el desarrollo de aplicaciones. Siguiendo el diagrama encontramos:

- **Activity Manager:** Administra la pila de actividades y gestiona el ciclo de vida de las aplicaciones en Android

- **Windows Manager:** Como su nombre indica, es el responsable de gestionar las ventanas y se encarga de decidir, que ventanas están visibles, y como se verán en la pantalla. También se encarga de la transición entre ventanas y animaciones cuando se abre y cierra una aplicación o cuando se rota la pantalla.

- **Content Provider:** Permite a las aplicaciones compartir y publicar información con las otras aplicaciones.

- **View System:** Conjunto de vistas que sirven para crear las interfaces de usuario.

- **Notification Manager:** Librería esencial que permite al usuario saber lo que está pasando a tiempo real. Ejemplos de eventos que conllevan una notificación son llamadas, mensajes, Wi-Fi disponibles, entre otras.

- **Package Manager:** El sistema por el cual las aplicaciones son capaces de encontrar información sobre otras aplicaciones instaladas en el dispositivo.

- **Telephony Manager:** Se encuentras las APIs encargadas de las funcionalidades propias del teléfono, llamadas, mensajes, etcétera. Provee la información a la aplicación sobre los servicios telefónicos disponibles, como por ejemplo el estado en el que se encuentran.

- **Resource Manager:** Librería que se encarga de dar acceso a las aplicaciones a recursos que no están en el código, como nombres, configuraciones de colores y capas de interfaces de usuario.

- **Location Manager:** Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles.

Librerías

El objetivo principal de esta capa es dar funcionalidad a las aplicaciones que lanzan tareas que se repiten con frecuencia, de tal modo que se evita la codificación en exceso y se garantiza una ejecución eficiente.
Entre las librerías más importantes en esta capa, se pueden encontrar las siguientes:

- **libc**: Librería para el lenguaje de programación C.
- **SQLite**: Se encarga de la creación y gestión de la base de datos.
- **SSL**: Responsable de las seguridad en internet.
- **Surface Manager**: Compone las ventanas que se muestran en la pantalla.
- **OpenGL/ES**: Encargada de dibujar los gráficos en 2D y 3D.
- **WebKit**: Asistente para navegadores.
- **FreeType**: El encargado de dar soporte para los distintos tipos de fuentes.
- **Media Framework:** Responsable de reproducir y grabar audio y vídeo.

Entorno de Ejecución de Android

Conjunto de bibliotecas base en las que se encuentran la gran mayoría de las funciones disponibles del lenguaje de programación Java y del lenguaje Android. Cada aplicación Android corre su propio proceso, con su instancia propia de la máquina virtual de Android.

La ventaja de utilizar la máquina virtual de Android es que las aplicaciones se compilan una única vez, y garantiza de que la aplicación funcionará en cualquier dispositivo Android que cumpla los requerimientos mínimos de versión.

Núcleo de linux

Android utiliza el núcleo de Linux 2.6. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.
Es considerado el corazón de la arquitectura Android, a parte de ser responsable de los drivers del dispositivo, también lo es del control de la batería, de la memoria, acceso a recursos y control del dispositivo.

3.1.4 Máquina virtual Android

Como se ha explicado de forma breve anteriormente, el sistema operativo Android utiliza una máquina virtual, que está situada en la capa de ejecución y su función principal es la de optimizar el uso de la memoria y los recursos del propio dispositivo móvil.

Hasta la versión 4.4 de Android, llamada KitKat, Android utilizaba la máquina virtual Dalvik, que esta basada en el uso de registros, ya que los teléfonos están optimizados para la ejecución de los mismos. Otra característica importante de Dalvik es que ha sido optimizada para que múltiples instancias de ella puedan funcionar al mismo tiempo con un impacto muy bajo en el rendimiento de la memoria del dispositivo. El objetivo de esto es proteger a las aplicaciones, de forma que el cierre o fallo inesperado de alguna de ellas no afecte de ninguna forma a las demás.

A partir de la versión 4.4, Android introduce una nueva máquina, llama ART. Ambas se encargan de compilar y traducir clases en formato bytecode a código máquina. La diferencia entre ART y Dalvik, es el momento en que realizan esta traducción. Dalvik utiliza “Just in time”, compila el código cada vez que se inicia una aplicación. Por el contrario, ART utiliza “Ahead of time”, que hace la compilación en el momento de la instalación, lo cual reduce el uso de los recursos al evitar la compilación mientras se está utilizando la aplicación.[21El coste, es que con la máquina ART, las aplicaciones tardan más en instalarse, alrededor de 30 segundos más, y ocuparan un 20 por ciento más de espacio en la memoria de nuestro teléfono[22].

3.1.5 Aplicaciones Android

Las aplicaciones Android están basadas en la combinación de componentes. Actualmente existen cuatro:

- Activities
- Services
- Broadcast receivers
- Content providers

Estos componentes tienen que ser declarados en el principal archivo de configuración de una aplicación Android, el manifiesto[23].

- **Activities:** Actividades que normalmente se realizan en primer plano y suelen tener una interfaz gráfica asociada. Puede haber varias actividades en una aplicación

- **Services:** Actividades que se ejecutan en segundo plano.

- **Broadcast receiver:** Se encargan de reaccionar frente a los anuncios que envía el mismo sistema como cambios en la batería, o anuncios de otras aplicaciones. Estos mensajes son enviados por difusión, por lo que los Broadcast receivers están siempre a la escucha.

- **Content provider:** Permite a las aplicaciones compartir y publicar información con las otras aplicaciones.

La activación de estos componentes es gracias a los Intents[24]. Los Intents son mensajes que se pueden utilizar para pedir una acción de un componente de otra aplicación. También facilitan las comunicación entre los componentes:

- Para empezar una Activity
- Para empezar un Service
- Para envíar un mensaje en modo broadcast

3.2 Introducción Android Wear

La definición más sencilla, un wearable es un dispositivo tecnológico que llevamos encima, estos dispositivos pueden ser relojes, gafas, pulseras, etcétera. En los últimos años se les ha dado funcionalidad, más concretamente en 2012. Es la fecha en la que Google sorprende al mundo con la demostración de las Google Glass[4], convirtiéndose en uno de los inventos de ese año y preparando el camino para lo que se avecinaba.

Lo que más sorprendió al público fue el tamaño de los wearables y toda la funcionalidad que cabía en ellos. Son pequeños pero no por ello menos poderosos, tienen características que parecen de ciencia ficción, que hace unos años parecían impensables.

En 2014 Google continúa por este camino liberando el SDK de Android Wear, planteando una nueva forma de ver los wearables. Se elimina la idea de que el wearable es un ente independiente que pretende sustituir al teléfono para convertirse en una extensión de éste, ampliando su funcionalidad.

Actualmente los Weareables disponen de una gran cantidad de sensores que ayudan a registrar nuestra actividad diaria. Sensores que, por ejemplo, pueden registrar nuestras pulsaciones en todo momento, tanto en reposo como durante una actividad física, pudiendo así prevenir problemas del corazón entre otras funciones.

La revolución de los wearables a la que estamos asistiendo es el resultado de llevar la tecnología a su tamaño más ínfimo, al desarrollo de baterías más eficientes y a la ampliación de las infraestructuras de comunicación.

Se puede afirmar, sin ningún miedo al error, que en la actualidad, llevamos más tecnología de la que podría tener un ordenador de sobremesa en los años 90.

Se prevé que las siguientes generaciones nunca vean los ordenadores del mismo modo en el que nosotros lo vemos ahora. En la actualidad, muchos usuarios ya han remplazado los ordenadores, tanto portátiles como de sobre mesa, por dispositivos móviles como tablets y teléfonos.

El alcance de los wearables no parece tener techo, por lo menos en un futuro cercano. Aunque en un primer momentos los wearables fueron concebidos como su nombre indica, una prenda de vestir con funcionalidad, cada vez son más los que los adquieren para razones médicas o deportivas.

Es muy probable que los wearables acaben siendo herramientas de trabajo, podemos imaginar pulseras enviando información de las constantes vitales de los trabajadores, enviando su geolocalización. Pacientes siendo monitorizados por pulseras y actualizando su propio historial de presión sanguínea y temperatura en el momento.

Los tres sectores en los que los wearables han tenido un gran impacto son:

- Fitness
- Relojes
- Gafas

3.2.1 Fitness

El uso de los sensores en los wearables, tales como pulseras de monitorización de actividad física, dispositivos de seguimiento GPS, han hecho que podamos tener un registro completo de nuestra actividad física. Años atrás, toda esta información era propia de deportistas de élite, y ahora puede ser accesible a cualquier persona con una pulsera y un teléfono gracias al uso de los sensores.

El uso inteligente de los sensores como acelerómetros, temperatura, GPS, giroscopios, medidores de pulsaciones, y muchos más hacen posible tener al instante información como:

- Temperatura
- Pulsaciones
- Presión sanguínea
- Pasos realizados
- Altitud
- Prevención de enfermedades
- Etcétera

Muchas de estas pulseras trabajan a través de las APIs, una serie de llamadas a funciones a las que se accede mayormente vía Bluetooth y de esta forma poder obtener la información que generan todos los sensores.

3.2.2 Relojes

Es un hecho que en la actualidad los relojes han sido de alguna forma apartados de la sociedad, en el sentido de que su función principal, que es mostrar la hora, nos la proporciona el teléfono.

Esto ha cambiado en los últimos años, se ha añadido funcionalidad extra al reloj, convirtiéndolo en un dispositivo inteligente, con una gran variedad de sensores, capaz de aportar mucha información a nuestra vida. Todos estos cambios han hecho que el reloj vuelva a recuperar el protagonismo perdido.

Los relojes son ahora unidades computacionales muy potentes, incluyen pantallas táctiles, sensores,

micrófonos y cámaras. Crear aplicaciones para estos dispositivos es más sencillo gracias a Android Wear que ha unificado los distintos SDKs que antes se necesitaban para su desarrollo.

3.2.3 Gafas

Sin duda el wearable que cautivó al mundo en aquella increíble demostración[1], se convirtió rápidamente en el dispositivo más interesante, también debido al hecho de que parecía completamente un objeto traído de un futuro lejano.

Las Google Glasses son la convergencia entre la realidad aumentada y la visión de un ordenador para convertirse en algo útil que se pueda usar en el día a día.

3.3 Google Play Services

3.3.1 Visión General

Las dos aplicaciones que se desarrollan para este trabajo de fin de grado, utilizan Google Play Services para conectarse entre ellas, por lo que es importante entender qué es antes de proseguir.

Google Play Services es un servicio que sirve para dar a tus aplicaciones más características para atraer a los usuarios. Utilizar Google Play te permite acceder a las características más potentes de Google, como por ejemplo Google Maps, Google Now y muchas más. De esta forma se ha conseguido integrar estos servicios de una forma muy fácil[25] .

3.3.2 Cómo funciona

Biblioteca cliente de Google Play Services

En esta biblioteca se encuentras las interfaces a los distintos servicios de Google y te permite obtener la autorización de lo usuarios para poder acceder a estos servicios con sus claves. También están las APIs[26] que te permite resolver cualquier problema en tiempo de ejecución, como por ejemplo, que este deshabilitada o se haya caducado el APK de Google Play Services.

Para tener acceso a los nuevos productos de Google, es tan fácil como actualizar a la nueva versión de la biblioteca del cliente. Si por el contrario, no te interesan las novedades, no es necesario que actualices, las funcionalidades que ya están en tu aplicación seguirán funcionando.

El APK de Google Play Services

El APK de Google Play Services contiene los servicios individuales de Google, y se ejecuta en segundo plano en Android. Para interactuar con el segundo plano se necesita la biblioteca cliente anteriormente mencionada. Los servicios llevan con ellos las acciones para usarlas en nuestro beneficio. Un procedimiento sencillo basta para obtener acceso a cada uno de los servicios de Google, lo que se traduce en una gran consistencia entre el desarrollador y los usuarios de la

aplicación.
Los dispositivos con la versión 2.3 de Android en adelante tienen Google Play Services instalado. Además recibe actualizaciones con gran frecuencia por lo que siempre están disponibles los nuevos productos de Google.

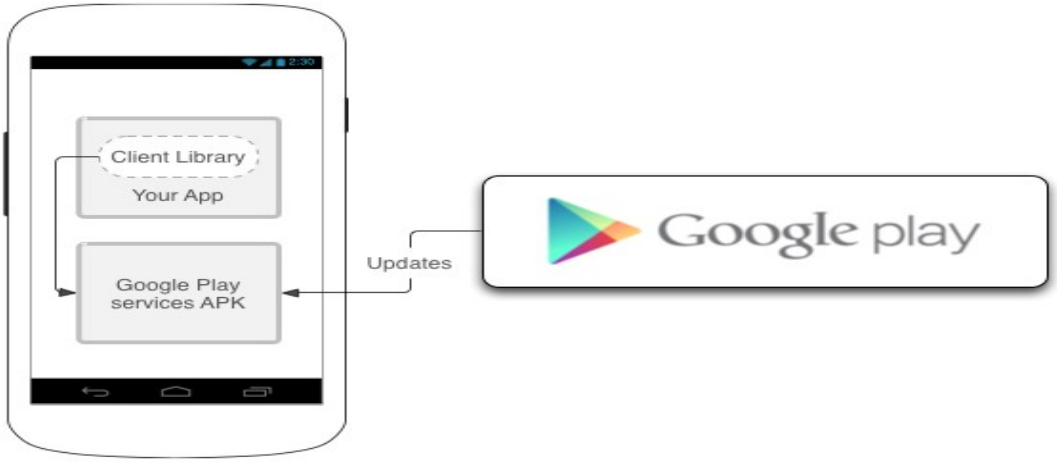

Figura 9. Funcionamiento de Google Play Services[25].

Los beneficios para tu aplicación

Google Play Services te da la libertad de usar las APIs más nuevas. Las actualizaciones se realizan de manera automática, por lo que el usuario no se tiene que preocupar por esto. Las nuevas versiones de la biblioteca del cliente son entregadas a través del Android SDK Manager.

También se pueden bloquear las notificaciones que nos parezcan molestas, esto es un punto importante, ya que en algunos momentos no es cómodo que el reloj este vibrando o luciendo cada vez que recibamos un aviso de alguna aplicación.

Cómo acceder a las APIs de Google

Cuando queremos crear una conexión con una de las APIs de Google de las que provee Google Play Services como Google Now o Google+ entre otros, es necesario crear una instancia de `GoogleApiClient`.

Este cliente provee un punto de entrada común a todos los servicios de Google y se encarga de controlar la conexión entre el dispositivo del usuario y cada servicio de Google.

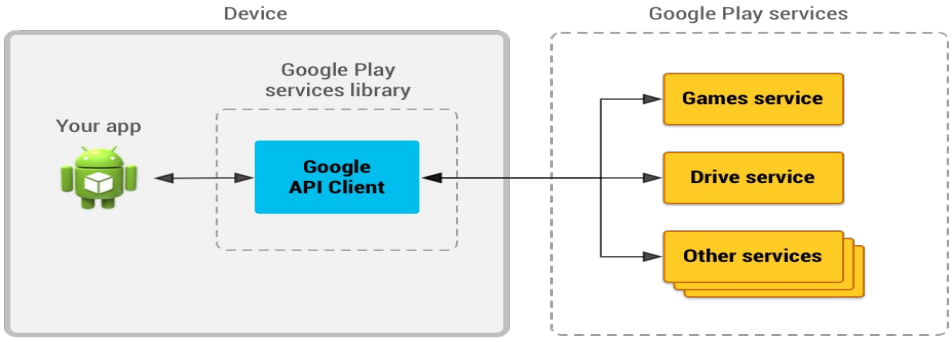

Figura 10. Cliente Google API que provee una interfaz para conectarse y hacer llamadas a los distintos servicios de Google [27].

Empezar una conexión

Una vez que el proyecto ya esta unido a los servicios de Google Play, hay que crear una instancia de `GoogleApiClient` usando las APIs de `GoogleApiClient.Builder` en el método `onCreate()` de tu actividad. Un ejemplo sencillo de conexión es el siguiente:

```
GoogleApiClient mGoogleApiClient = new GoogleApiClient.Builder(this)
    .addApi(Drive.API)
    .addScope(Drive.SCOPE_FILE)
    .build();
```

Bloque de código 1. Ejemplo de conexión con `GoogleApiClient`[28].

Es importante destacar, debido a que si se añade el API de `Wearable` junto con otras APIs de `GoogleApiClient`, como es el caso de las dos aplicaciones que se han desarrollado para este trabajo de fin de grado, podemos tener errores de conexión en los dispositivos que no tengan la aplicación de Android Wear instalada. Para evitar estos errores, hay que hacer uso de la función `addApiIfAvailable()` e indicarle el API `Wearable` que se va a usar, de esta forma debería manejar correctamente que falte esta API.

```
addApiIfAvailable(Wearable.API).build();
```

Bloque de código 2. Ejemplo de llamada al método `addApiIfAvailable()`.

Antes de empezar cualquier conexión llamando al método `connect()` es necesario especificar una implementación para las llamadas de respuesta y los fallos de conexión, `ConnectionCallBacks` and `OnConnectionFailedListener`. Estas interfaces reciben las llamadas de respuesta con los resultados de la llama al método `connect()`, esta respuesta

indica como ha ido la conexión con Google Play Services y puede ser, satisfactoria, fallida o suspendida.

```
 private GoogleApiClient apiClientFactory() {
        return new GoogleApiClient.Builder(getApplicationContext())
.addConnectionCallbacks(new GoogleApiClient.ConnectionCallbacks() {
.
.
.
```

Bloque de Código 3. Ejemplo de uso de `ConnectionCallBacks`. Extracto de código de Find Your Phone

Una vez que se han definido estan interfaces, ya se puede llamar al método `connect()`. Para manejar de forma correcta el ciclo de vida de la conexión, se debe llamar a `connect()` en el método `OnStart()` de la actividad, después llamar a `disconnect()` en el método `onStop()`.

3.4 Aplicación móvil Android Wear

Los dispositivos móviles Android disponen desde hace ya algún tiempo una aplicación llamada Android Wear.

Esta aplicación es el nexo de unión entre el teléfono y el reloj. A través de ésta podemos conectar ambos dispositivos a través del bluetooth de éstos. La mayor limitación de esta forma de conexión es la distancia a la que tienen que estar el uno del otro. Si se sobre pasa esta distancia se rompe la comunicación.

Si ésta se rompe, el reloj vibrará durante unos segundos y nos preguntará si nos hemos olvidado el teléfono, lo cual puede llegar a ser bastante útil.

Desde la aplicación se pueden cambiar las distintas “caras del reloj” (Watch faces), se pueden gestionar las aplicaciones[29].

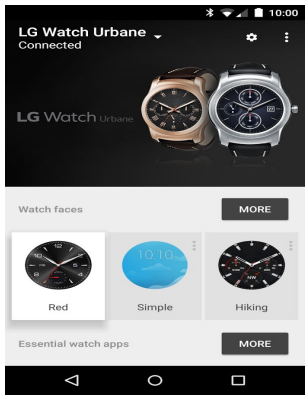

Figura 11. Aplicación Android Wear[29].

3.5 Desarrollo de aplicaciones para Android Wear

3.5.1 Notificaciones

3.5.1.1 Introducción

Las notificaciones son una de las formas esenciales de decirle al usuario que está pasando en cada momento. Nos informan de los eventos que ocurren, como llamadas, mensajes, etcétera. En este último tiempo las notificaciones han pasado de ser un simple mensaje en la pantalla, a convertirse en elementos interactivos. El hecho de que sean interactivas e incluyan diferentes acciones hacen que la interacción entra el usuario y la aplicación sea mucho más fácil.

Cuando un dispositivo Android, ya sea un teléfono o una tablet y un dispositivo Android Wear se conectan, el teléfono comparte automáticamente las notificaciones con el reloj.
En el reloj, cada notificación aparece como una nueva tarjeta en un `GridView` en vez de una `ListView` como es el caso del teléfono.

El `GridView` muestra la información en panel en dos dimensiones, de tal forma que las notificaciones quedan desplegadas en forma matricial en la que es posible desplazarse de una a otra mediante el desplazamiento del dedo. Por el contrario `ListView` muestra las notificaciones en formato lista.

Con Android Wear, Google ha mantenido su estrategia, las notificaciones juegan un papel muy importante en cómo el usuario interactúa con la aplicación, mediante lo que Google llama micro interacciones. Cada notificación en Android Wear puede consistir en varias subnotificaciones llamadas páginas. Cada página es una pequeña parte de la experiencia de la notificación, como la interacción, la información o la retroalimentación.

En este apartado vamos a entrar un poco más en detalle en el mundo de las notificaciones.
Puede haber notificaciones con las siguientes acciones entre otras:

- Contestar desde el reloj mediante la voz.
- Indicar desde el reloj que queremos abrir la notificación desde el teléfono.
- Leer la notificación desde el reloj y no realizar ninguna acción.
- Bloquear las notificaciones de esa aplicación en el reloj.

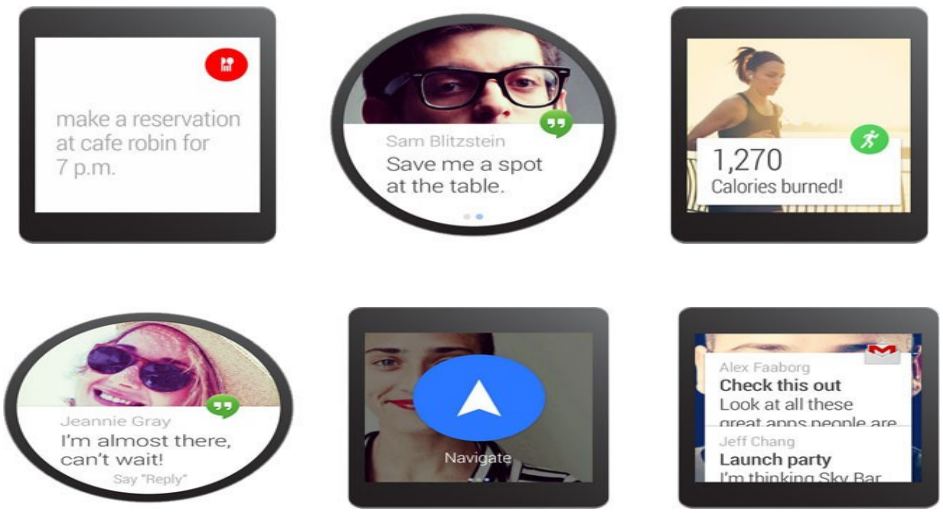

Figura 12. Diferentes tipos de notificaciones en Android Wear.

3.5.1.2 Diseño y creación de una notificación

Las notificaciones son una parte muy importante de la interfaz de usuario, y por ello tienen que tener su propia guía de diseño. Los cambios introducidos en el material de diseño en Android 5.0 son de gran importancia, y hay que tenerlos en cuenta a la hora de crear notificaciones[31].

Es importante saber qué es obligatorio y qué no cuando estamos desarrollando notificaciones. En la clase Android `Notification.Builder` podemos encontrar los siguientes métodos:

- `setContent(RemoteViews)`
- `setContentInfo(CharSequence)`
- `setContentIntent(CharSequence`
- `setContentText(CharSequence)`
- `setContentTitle(CharSequence)`
- `setLargeIcon(Bitmap)`
- `setPriority(int)`
- `setSmallIcon(int)`
- `setStyle(Notification.Style)`
- `setWhen(long)`

De todos estos, solo el método `setSmallIcon(int)` es obligatorio a la hora de crear notificaciones para el teléfono, el resto son opcionales.

En cuanto a Android Wear, incluye los mismos métodos y añade nuevas características solo para wearables:

- `addAction(Action)`
- `addPage(Notification)`

- `setBackground(Bitmap)`
- `setContentAction(int)`
- `setGravity(int)`
- `setHintHideIcon(boolean)`

En el caso de los wearables, los métodos obligatorios para la creación de las notificaciones son:

- `setSmallIcon()` Para añadir un icono pequeño
- `setContentTitle()` Para poner un título
- `setContentText()` Para añadir un texto

Para obtener más información de la clase `Notification.Builder` y ver el resto de métodos, hay que consultar la información sobre la clase[30].

Para usar las características específicas de las notificaciones de Android Wear, hay que heredar las clases `WearableExtender y Notification.Builder` una vez hecho esto, ya podemos usar cualquiera de los métodos anteriormente citados y muchos más para personalizar la notificación a nuestro gusto. Un pequeño ejemplo de la creación de una notificación sería tal que así:

```
NotificationCompat.WearableExtender wearFeatures = new

NotificationCompat.WearableExtender();

Notification notification = new NotificationCompat.Builder(this)
.setContentTitle("Ejemplo de notificación")
.setContentText("El texto que llevará la notificación!")
.extend(wearFeatures)
.build();
```

Bloque de código 4. Creación de una notificación simple.

3.5.1.3 Recibir entradas de voz en una notificación

El usuario tiene la posibilidad de responder a una notificación directamente a través de la voz. Alternativamente se pueden crear unas entradas de texto predefinidas para agilizar la respuesta[32].

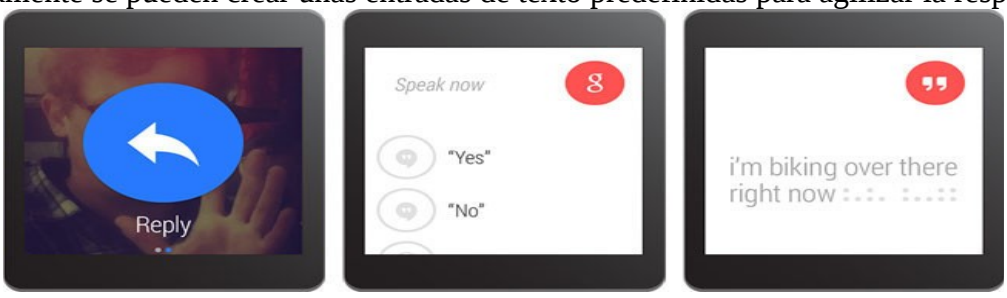

Figura 13. Distintos pasos para responder mediante la voz.

La primera imagen corresponde con la acción de responder, seguidamente se podría elegir de manera táctil una de las respuestas predefinidas, o directamente dictando la respuesta[33]. Cuando el usuario responde ya sea por voz o con una de las respuestas predefinidas, el sistema envía un mensaje a la aplicación conectada con el teléfono. El mensaje se adjunta en un `Intent` y se envía.

3.5.2 Acciones de voz

3.5.2.1 Introducción

Las acciones que se llevan a cabo a través de la voz son de vital importancia en este tipo de dispositivos. Hay que recordar que el smartwatch tiene un tamaño similar a un reloj de muñeca y no dispone de teclado, por lo que la comunicación mediante la voz entre el reloj y el usuario facilita mucho la interacción entre ambos.

Existen varios tipos de acciones por voz. Puedes decirle a tu reloj comandos simples, y éstos podrán ser reconocidos como una serie de comandos predefinidos. Para ello, el sistema compara la entrada de voz, con una serie de comandos en una tabla, en caso de existir, la acción se llevará a cabo en el momento[34].

Otra de las opciones es enviar directamente el audio generado en el reloj, al teléfono o a la tablet a la que esté conectada y que sea esté el encargado de decodificarlo. Android Wear dispone de dos tipos de acciones de voz:

- Las que provee el sistema
    - Estas acciones están basadas en tareas y están desarrolladas en la plataforma Wear. Se filtran dependiendo de la actividad que queramos lanzar cuando se ha capturado la voz. Algunos ejemplos que ya vienen instalados en el reloj son “Crear una nota” o “Configurar la alarma”
- Las que proveen las aplicaciones
    - Son acciones de voz que se basan en las aplicaciones, y son declaradas igual que un icono de una aplicación. Para usarla el usuario dice “Empezar ...” y la actividad que especifiques empezará.

3.5.2.2 Tipos de acciones

Android Wear dispone de dos tipos de acciones de voz:

- Las que provee el sistema
    - Estas acciones están basadas en tareas y están desarrolladas en la plataforma Wear. Se filtran dependiendo de la actividad que queramos lanzar cuando se ha capturado la voz. Algunos ejemplos que ya vienen instalados en el reloj son “Crear una nota” o “Configurar la alarma”

- Las que proveen las aplicaciones
    - Son acciones de voz que se basan en las aplicaciones, y son declaradas igual que un icono de una aplicación. Para usarla el usuario dice “Empezar ...” y la actividad que especifiques empezará.

Declarar acciones de voz del sistema

La plataforma Android Wear provee bastantes comandos por voz que están basados en las acciones más básicas que puede hacer un usuario como “Tomar una nota” o “Configurar la alarma”. Esto permite al usuario decir lo que quiera hacer y dejar al sistema que se encargue de elegir cual es la mejor actividad para empezar. Cuando el usuario dice la acción mediante la voz, la aplicación filtra para saber qué acción tiene que ser lanzada.

Si quieres empezar un servicio para hacer algo en segundo plano, hay que mostrar una actividad como una señal visual y empezar el servicio en la actividad. Asegurarse de llamar a la función `finish()` cuando queramos deshacernos de la señal visual.

Declarar acciones de voz para aplicaciones

Si ninguna de las acciones de voz de la plataforma no encajan para ti, puedes arrancar las aplicaciones directamente con una acción de voz que diga lo siguiente por ejemplo “Empezar NombreDeMiAplicacion”.

Registrar una acción de “Empezar” es lo mismo que registrar un icono en el teléfono. Pero al contrario de pedirte un icono en un lanzador, la aplicación requiere una acción de voz.
Con cualquiera de las entradas de voz, el teléfono espera una respuesta en formato de texto.

Si enviamos la información en el sentido reloj-teléfono, es el reloj el que se encarga de recuperar el mensaje de voz, transformarlo en texto, y enviarlo al teléfono. En el Bloque de código 2, se aprecia, que cuando la actividad devuelve el resultado, dependiendo del contenido de el mensaje se puede hacer una cosa u otra.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE)
   {
         if (resultCode == RESULT_OK) {

              ArrayList<String> textMatchList =
              data.getStringArrayListExtra
              (RecognizerIntent.EXTRA_RESULTS);

              if (!textMatchList.isEmpty()) {
                  String voiceMessage = StringUtils.join
                  (textMatchList.iterator(), "#");
                  voiceMsg = voiceMessage;
              }
              if(voiceMsg.equalsIgnoreCase("facebook")){
                  openFacebook();
              }
              if(voiceMsg.equalsIgnoreCase("twitter")){
                  openTwitter();
              }
              if(voiceMsg.equalsIgnoreCase("instagram")){
                  openInstagram();
              }
              Wearable
         .MessageApi
         .sendMessage(apiClient,remoteNodeId, voiceMsg, voiceMsg.getBytes());
          }

      }
      super.onActivityResult(requestCode, resultCode, data);
  }
```

Bloque de código 5. Ejemplo de filtro en el reloj en Find Your Phone.

Los tipos de interacciones a través de la voz también se pueden estructurar como indica la siguiente tabla:

Interacción	**Descripción**
Enviar un comando a un host	El dispositivo ejecuta la operación de reconocimiento de voz y envía un comando al dispositivo host. En la documentación oficial, son las acciones de voz basadas en aplicaciones.
Enviar un mensaje al host	El dispositivo Wear captura el audio y lo envía para producir un stream de bytes de audio para que pueda ser enviado al dispositivo host.
Responder a una petición del host	Cuando llega una notificación, es posible obtener un conjunto de respuestas predefinidas a las cuales el usuario puede responder mediante la voz.
Responder a una petición del host en un mensaje de texto	Cuando llega una notificación como por ejemplo, un e-mail, es posible responder directamente hablando al dispositivo wearable y enviar esta respuesta en formato de texto como respuesta.
Ejecutar comandos simples del sistema	El dispositivo wearable ejecuta la operación de reconocimiento de voz basada en una serie de intents predefinidos. En la documentación oficial, son las acciones de voz del sistema.

Tabla 1. Distintos tipos de acciones de voz.[35]

3.5.2.3 Diseño de una acción de voz

Tomando como ejemplo la entrada de voz realizada en la aplicación Social Media Opener, Bloque de código 3, donde se hace uso de dos métodos, en el primero se comprueba que el dispositivo tenga reconocimiento de voz y en el segundo se lanza la actividad con una serie de características especificadas por nosotros. Como el lenguaje, el número de resultados, etcétera. Todas estas opciones estan definidas en el paquete Android `android.speech`.

```
public void checkVoiceRecognition() {
      PackageManager pm = getPackageManager();
      List<ResolveInfo> activities = pm.queryIntentActivities
     (new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
      if (activities.size() == 0) {
          mbtSpeak.setEnabled(false);
          showToastMessage("Voice recognizer not present");
      }
  }
```

Bloque de código 6. Método que se encarga de comprobar si el dispositivo dispone de reconocimiento de voz.

```
public void speak(View view) {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, getClass()
            .getPackage().getName());

    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_WEB_SEARCH);
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 1);
    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
```

Bloque de código 7. Método necesarios para la configuración de la entrada de voz.

3.5.3 Envío y sincronización de información.

3.5.3.1 Introducción

Esta capa de la API es fundamental, es la que hace posible el intercambio de información entre el teléfono y el reloj y viceversa.

Dentro de la API, la capa de información del reloj, es parte de los servicios de Google Play y provee de un canal de comunicación entre el teléfono y las aplicaciones del wearable. La API consiste en un conjunto de objetos con información que el sistema puede enviar y sincronizar por la red, y escuchadores que notifican a tus aplicaciones de los eventos importantes con la capa de información[38].

3.5.3.2 Elementos para enviar y recibir la información.

Los elementos que forman parte del sistema de comunicación son:

Data Items

Un `DataItem` provee de almacenamiento de información con sincronización automática entre el teléfono y el reloj.

Messages

La clase `MessageApi` puede enviar mensajes y es buena para las llamadas del tipo RPC (Remote Procedure Calls). Los ejemplos de uso más comunes son controlar el reproductor de vídeo del teléfono desde el reloj o lanzar un intent en el reloj desde el teléfono.

Los mensajes también son importantes para las peticiones en un sistema de comunicación de una sola dirección o para un modelo de petición y respuesta.

Si ambos dispositivos están conectados el sistema encola los mensajes para ser enviados y devuelve un código de resultado de éxito. Hay que destacar que el código de éxito no significa que se haya entregado con éxito pues los dispositivos se pueden desconectar durante el proceso.
Si los dispositivos no están conectados, el sistema devuelve un código de error.

Asset

Los objetos `Assets` son para enviar bloques binarios de información, como imágenes. Se adjuntan `Assets` a objetos de información y el sistema automáticamente se encarga de enviarlo por el usuario, conservando el ancho de banda del bluetooth cacheando `Assets` de gran tamaño para evitar la retransmisión de información.

WearableListenerService (para los servicios)

Heredar de `WearableListenerService` te permite escuchar eventos importantes en la capa de información en una servicio.

El sistema se encarga del ciclo de vida del `WearableListenerService`, conectando con el servicio cuando éste necesite enviar objetos con información o mensajes y desconectando el servicio cuando no se necesite hacer ninguna tarea.

DataListener (activities en primer plano)

Implementar la clase `DataListener` en una actividad te permite escuchar eventos importantes que ocurren en la capa de información cuando la actividad esta en primer plano.

Usar esta clase en vez de `WearableListenerService` te permite escuchar los cambios solo cuando el usuario esta activamente usando la aplicación.

Channel

La clase `ChannelApi` se usa para enviar grandes bloques de información, como musica o películas, del teléfono al dispositivo wearable.

La API de `Channel` utilizada para enviar grandes bloques tiene las siguientes ventajas:

- Enviar grandes archivos de datos entre dos o más dispositivos conectados, sin la sincronización automática que se provee cuando se utilizan objetos Assets adjuntos a objetos `DataItem`. La API de `Channel` ahorra espacio en disco, no como la clase `DataApi`, que crea una copia de los `Assets` en el dispositivo de forma local antes de la sincronización con los dispositivos conectados.

- Envío fiable de un fichero que es demasiado grande en cuanto a tamaño para enviarlo usando la clase `MessageApi`

- Enviar datos en streaming, como música obtenida de un servidor de red o voz de un micrófono.

Hay que tener en cuenta que estas APIs han sido diseñadas para la comunicación entre teléfonos/tablets y wearables, y son las que tienen que ser usadas para establecer la comunicación entre estos dispositivos. Por ejemplo, no se debe intentar abrir una conexión de bajo nivel con sockets para crear un canal de comunicación.

Android Wear soporta una conexión múltiple entre varios wearables conectados a un dispositivo móvil de mano (teléfono o tablet). Por ejemplo, cuando el usuario guarda una nota en el teléfono, ésta automáticamente aparece en los wearables del usuario.

Para sincronizar la información entre los dispositivos, los servidores de Google disponen de una nube en la red de dispositivos.

El sistema sincroniza la información directamente con los dispositivos conectados, el nodo que esta en la nube, y los dispositivos wearables conectados al nodo en la nube a través de la red Wi-Fi.

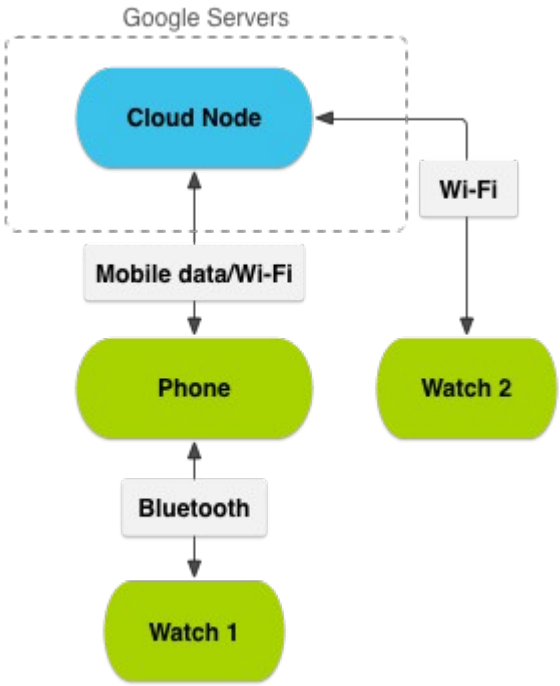

Figura 14. Sistema de comunicación teléfono, reloj y la nube.

La Figura 14 ilustra perfectamente el sistema de comunicaciones que utilizan estos dispositivos. El teléfono se conecta a la nube mediante datos móviles o Wi-Fi, y éste a su vez se conecta al reloj mediante bluetooth.

De tal modo, que el reloj siempre se encontrará conectado a internet a través de la conexión Bluetooth con el teléfono.

3.5.3.3 Diseño de comunicación entre dispositivos

Para que sea posible la comunicación entre los dispositivos, necesitamos una clase que se encargue de escuchar los eventos, tanto en el módulo Wear, como en el del teléfono. Esta clase tiene que heredar de la clase `WearableListenerService`.

Es necesario abrir un cliente de Google, que será el encargado de establecer la comunicación. Una vez establecida esta comunicación, necesitamos una función que se encargue de buscar los nodos conectados al cliente de Google Play Services, una vez que tenemos estos nodos, ya puede comenzar el envío de información entre el reloj y el teléfono. Se explicarán estas funciones en detalle en las secciones de las aplicaciones propias de esta memoria.

4. Configuración del entorno y primeros pasos

En este apartado se va a hacer un pequeño tutorial introductorio a la programación en el entorno de desarrollo elegido por Google para desarrollar aplicaciones Android, Android Studio.

4.1 Introducción

Hasta Febrero de 2015 el entorno por excelencia para desarrollar aplicaciones Android era Eclipse, pero en esa fecha Android/Google deciden evolucionar Android Studio de su versión Beta a una versión estable.

Otro de esos anuncios fue que el entorno de Eclipse dejaría de tener soporte, por lo que todos los desarrolladores han tenido que hacer el cambio.

Android Studio es un entorno orientado exclusivamente a la programación en este lenguaje, lo que supone numerosas ventajas, como el hecho de que el importar y gestionar librerías sea bastante más fácil que con Eclipse.

Debido a que Android Studio se ha convertido en el IDE oficial para el desarrollo de aplicaciones Android, vamos a entrar un poco en detalle de que es lo que nos ofrece[36]:

- Desarrollo flexible basado en Gradle
- Distintas variantes de desarrollo y generación múltiple de ficheros apk
- Plantillas de ejemplo que te ayudan a desarrollar características comunes de las aplicaciones
- Un editor de capas que con sistema arrastra y suelta (Drag and drop)
- Herramientas lint para detectar código sospechoso, rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Y muchas más..

4.2 SDK Manager

El SDK Manager de Android separa las herramientas del SDK, plataformas y otros componentes dentro de paquetes para tener fácil acceso y manejo. Se puede configurar para que automáticamente busque actualizaciones y nos notifique cuando un paquete se ha actualizado, una vez recibida la notificación, se pueden actualizar los paquetes accediendo directamente desde la notificación.

Una vez descargado el entorno Android Studio[DownLoad], lo primero es actualizar Android SDK. En las opciones del entorno, *Tools → Android → SDK Manager*

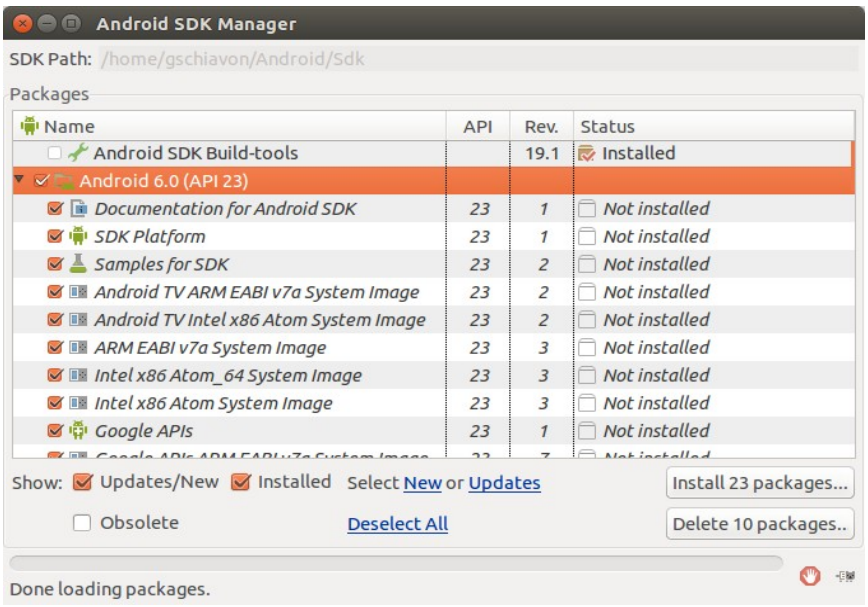

Figura 15. Android SDK Manager

En la figura 15 podemos instalar todos los paquetes necesarios para el correcto funcionamiento de nuestra aplicación.

4.3 Creación de un proyecto y primeros pasos

Las aplicaciones de los wearables se ejecutan directamente en el dispositivo, dándote acceso directo al hardware, como los sensores. Básicamente es lo mismo que cualquier aplicación desarrollada para un dispositivo Android usando el SDK de Android, pero difieren ampliamente en el diseño, la usabilidad y la cantidad de funcionalidad que se puede añadir a los wearables. Estas son las diferencias principales entre las aplicaciones para un teléfono y un wearable:

- Las aplicaciones para wearables son relativamente pequeñas en tamaño comparadas con las aplicaciones para el teléfono. Estas contienen solo lo que tiene sentido para un wearable, lo que suele ser, un pequeño conjunto de lo correspondiente para un teléfono. En general, se deberían hacer las operaciones más costosas en el teléfono cuando sea posible y enviar los resultados al wearable.

- Los usuarios normalmente no descargan las aplicaciones directamente desde el wearable. Se conecta la aplicación wearable con el teléfono dentro de la aplicación Android Wear de éste. Cuando el usuario instala una aplicación que contiene un módulo para wearable, de manera transparente para el usuario, la aplicación Android Wear automáticamente instala la aplicación en el reloj. Sin embargo, para propósitos de desarrollo, se puede instalar la aplicación directamente en el wearable.

Primeros pasos y requisitos

- Un dispositivo Android Wear
- Entorno de desarrollo Android Studio

Paso 1: Crear un proyecto

File → New → New Proyect

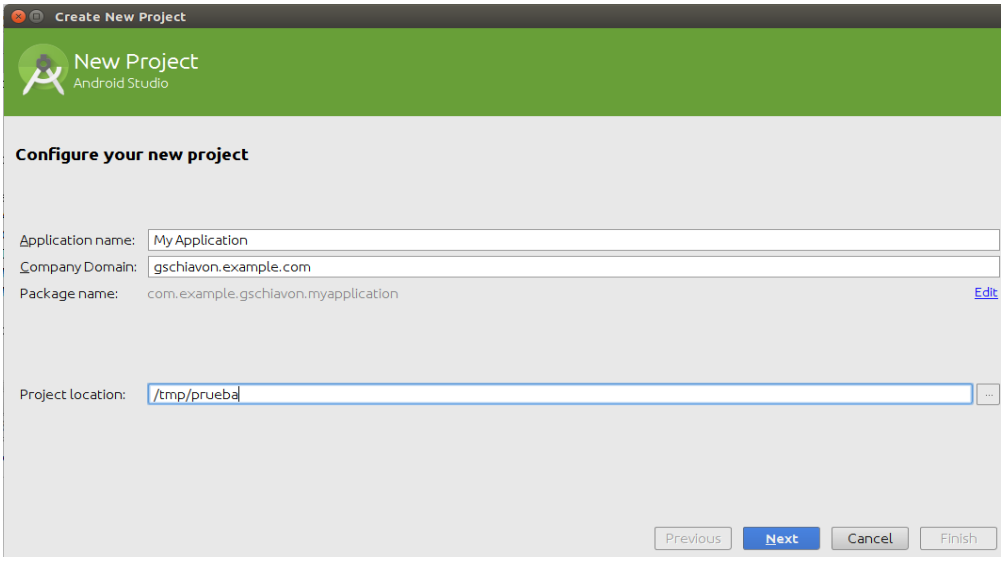

Figura 16. Crear proyecto nuevo.

Después de hacer *click* en “next” hay que seleccionar el módulo de “Phone and Tablet” y “Wear”

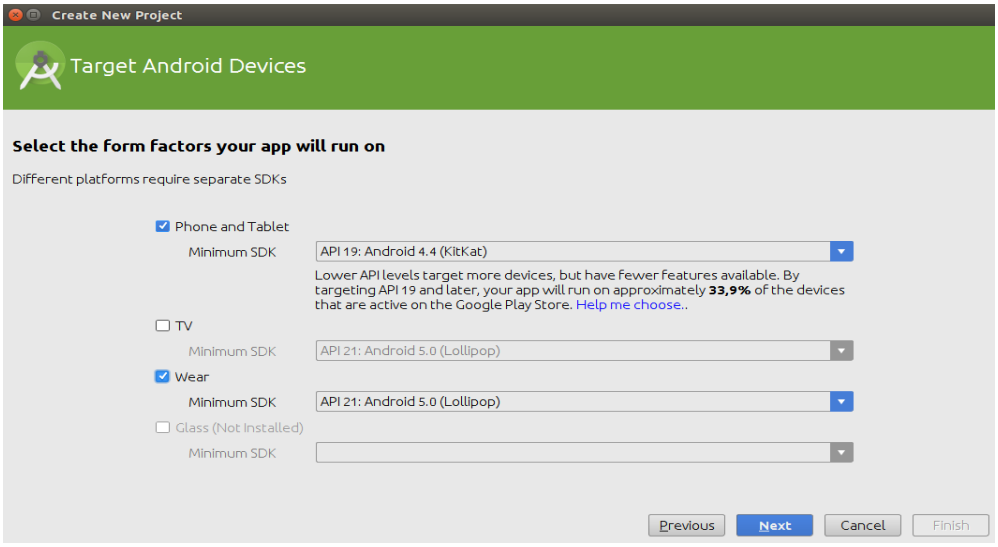

Figura 17. Seleccionar módulos del proyecto.

En el módulo wear tenemos que seleccionar la actividad con la que queremos que empiece nuestra aplicación:

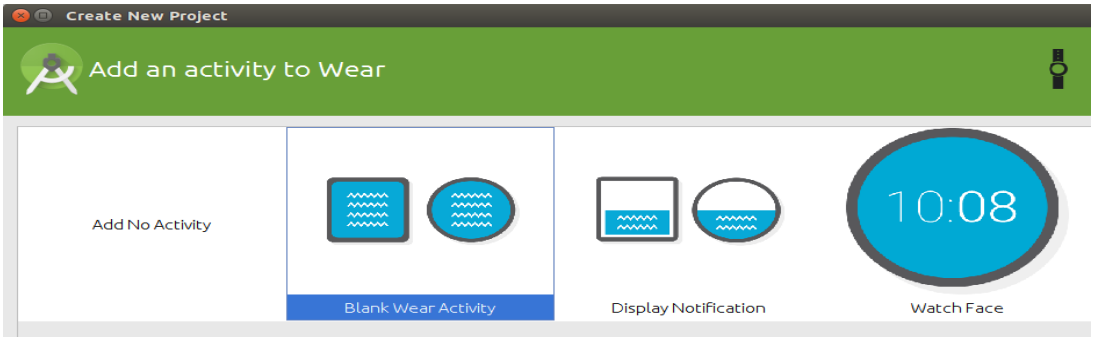

Figura 18. Distintos tipos de actividades para empezar

En este caso seleccionamos una actividad en blanco, y posteriormente le damos un nombre a nuestra clase principal.
Una vez hecho esto, el directorio del proyecto tiene que quedar de esta forma:

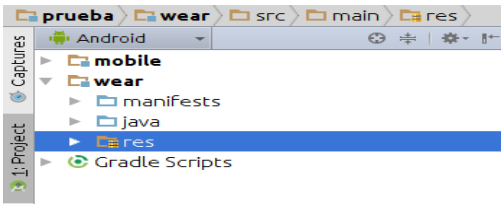

Figura 19. Directorio del proyecto.

De tal modo que haya dos módulos, mobile y wear. Como sus nombres indican, toda la parte relacionada con la programación en el reloj tiene que ir dentro de la carpeta “java” dentro del módulo wear, y análogamente en la carpeta mobile para el teléfono.

En la figura 20, está seleccionada la carpeta “res”, que es donde se encuentran todos los recursos que vayamos a usar para nuestra aplicación, como iconos, nombres, etcétera. La carpeta “res” desplegada tiene el aspecto que se muestra en la figura 20.

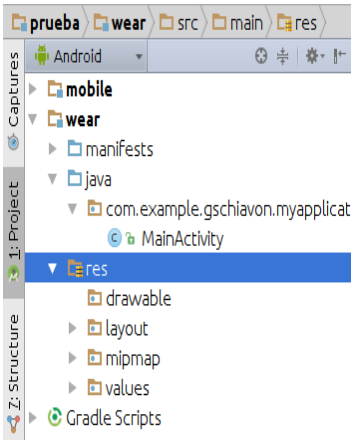

Figura 20. Carpeta “res” desplegada.

El sistema de construcción de Android es Gradle, un sistema que coge las mejores funcionalidades de otros sistemas de construcción y los combina[39]. Es un sistema basado en plugins, lo que significa que puedes automatizar las tareas de crear paquetes como jars y después poder escribir plugins en Java o Groovy y distribuirlos al resto del mundo.

Hay tres archivos Graddle, uno para el proyecto, otro para el módulo de móvil, y otro para el de wear. El del módulo wear es de esta forma:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 22
    buildToolsVersion "22.0.1"

    defaultConfig {
        applicationId "com.example.gschiavon.myapplication"
        minSdkVersion 21
        targetSdkVersion 22
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.google.android.support:wearable:1.1.0'
    compile 'com.google.android.gms:play-services-wearable:7.3.0'
}
```

Figura 21. Graddle del módulo wear.

En este caso se puede ver que en las “dependencies” hay un paquete de Google Play Services. En esta sección es donde incluiríamos todos los plugins que se utilizan en el módulo de wear.
También podemos ajustar las versiones en caso de que fuera necesario. Hay que destacar que este fichero es autogenerado, solo hay que añadir las dependencias propias de nuestro proyecto.
Una vez terminado este proceso de configuración ya estamos preparados para empezar a programar.

4.4 Emulador

El emulador es una herramienta importantísima a la hora de probar nuestras aplicaciones. Como su propio nombre indica es capaz de emular distintos dispositivos, en la figura 22 se aprecian los distintos dispositivos que soporta:

- Televisión
- Teléfono móvil
- Reloj inteligente
- Tablet

Para configurar el emulador hay que navegar hasta “AVD Manager”, *tools → Android → AVD Manager*

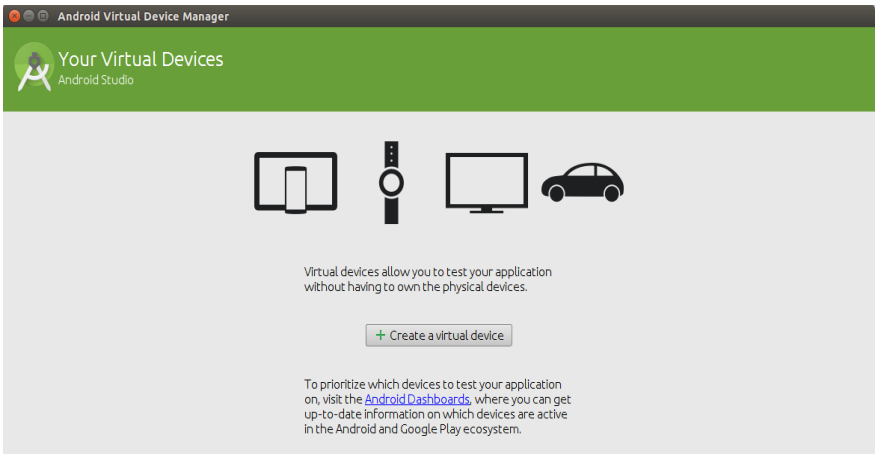

Figura 22. Distintos dispositivos para emular.

Después de hacer en click en “create virtual device”, ya podemos elegir cual queremos

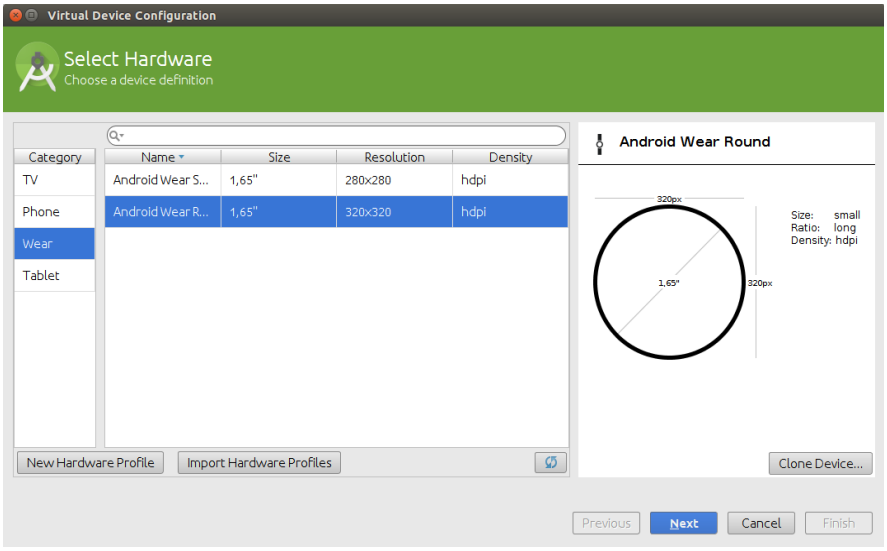

Figura 23. Selección de un simulador de reloj inteligente.

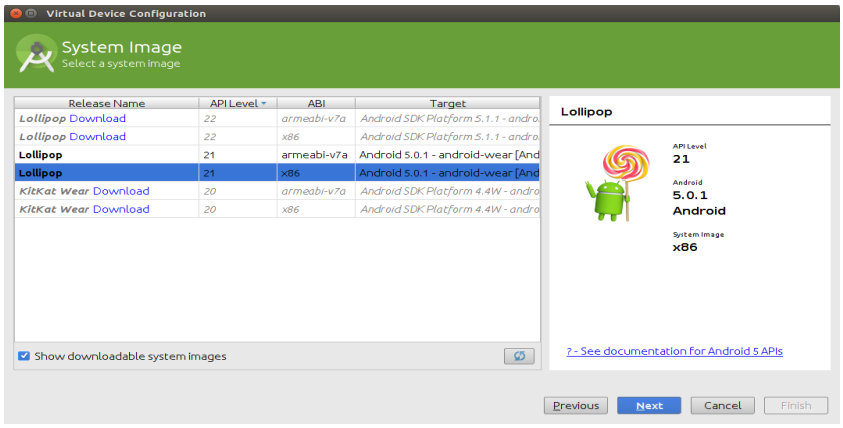

Figura 24. Selección de la versión de Android.

Una vez que tenemos creado nuestro emulador de un reloj inteligente. Situamos el lanzador en el módulo wear como indica la figura 25.

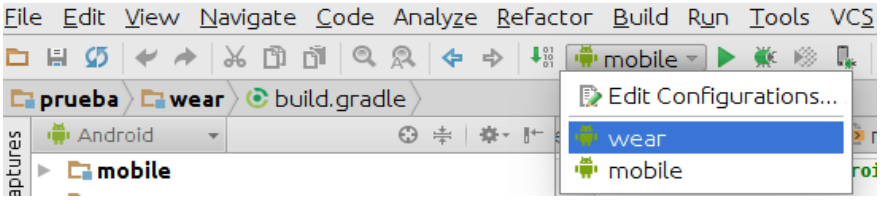

Figura 25. Seleccionamos el módulo wear.

Pulsamos el botón de “play” y seleccionamos para que ejecute el emulador.

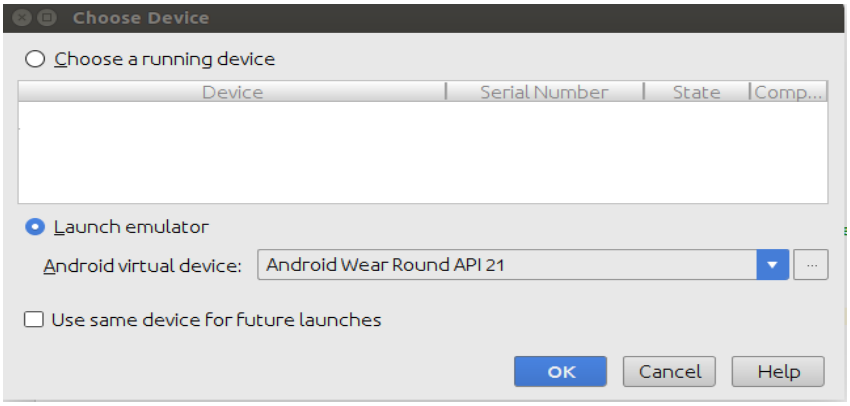

Figura 26. Seleccionamos que lance el emulador.

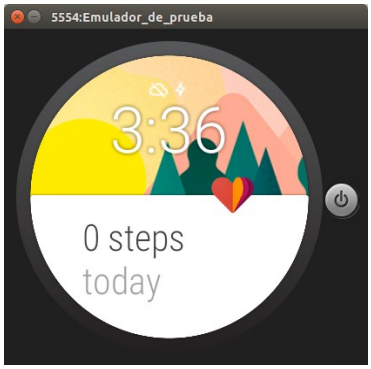

Figura 27. Emulador lanzado.

En la figura 27 ya se puede ver que se ha ejecutado correctamente el emulador y ya está listo para probar nuestras aplicaciones.

Si queremos probar la comunicación entre el teléfono físico y el dispositivo virtual tenemos que hacer lo siguiente:

- Conectar el teléfono al ordenador mediante USB
- Conectar el emulador al teléfono:

-Windows

```
cd C:\Android\sdk\platform-tools

adb.exe devices

adb.exe -d forward tcp:5601 tcp:5601
```

Figura 28. Conectar el emulador al teléfono.

-Linux

```
sudo ./adb kill-server
sudo ./adb start-server
sudo ./adb devices
```

Figura 29. Conectar y comprobar los dispositivos conectados.

Una vez que tenemos el teléfono y el emulador conectado, ya podemos probar las aplicaciones en los dos dispositivos.
Hay que tener en cuenta las limitaciones del emulador, para ello, podemos ver los detalles de nuestro dispositivo virtual para ver exactamente sus características, como sus sensores:

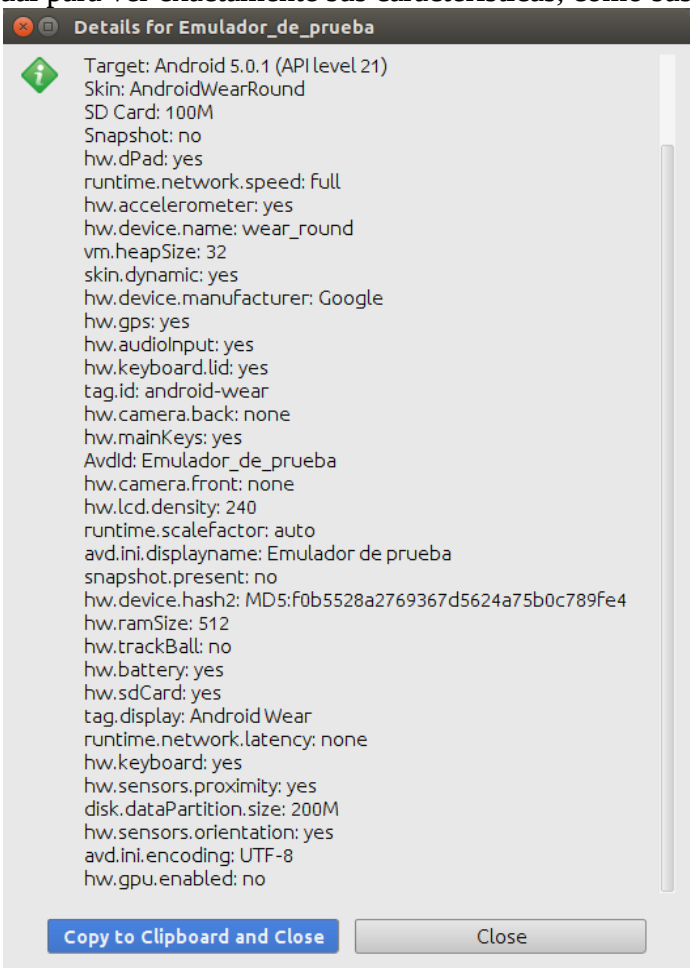

Figura 30. Características del dispositivo creado por el emulador.

4.5 Activar el reloj inteligente en modo desarrollador

Para poder utilizar nuestro reloj para probar las aplicaciones, es necesario activarlo en modo desarrollador[40]. Para hacerlo, tenemos que ir a la configuración del teléfono, pulsar la opción “Acerca de” y pulsar 7 veces encima de la zona del “Número de serie”, una vez hecho esto, saldrá un mensaje notificando que ya estamos en modo desarrollador.

Una vez activado este modo, tendremos activas las opciones ocultas de desarrollador, que están en el menú de configuración, “Opciones de desarrollo”. Entramos en las opciones y activamos la depuración ADB como muestra la figura 31.

Figura 31. Activar el modo depuración ADB.

4.6 Conectar el reloj inteligente al teléfono

Para conectar el teléfono y el reloj se tiene que hacer a través de la aplicación Android Wear del teléfono. Para ello hay que activar el modo bluetooth del teléfono y esperar a que la aplicación encuentre la señal bluetooth del reloj. Una vez conectados, automáticamente empiezan a compartir las notificaciones.

5. Aplicación Find Your Phone

En este capítulo se explicarán las funciones básicas de esta aplicación y se analizará en detalle el desarrollo técnico realizado.

5.1 Alcance y Objetivos

La aplicación Find Your Phone tiene una funcionalidad muy específica, que es la de encontrar el teléfono con el que está conectado el reloj mediante bluetooth. La mayor restricción que tiene la aplicación es la distancia a la que puede estar el reloj del teléfono para que sigan conectados. El bluetooth tiene un alcance máximo de unos diez metros aproximadamente, por lo que si el teléfono se encuentra a una distancia superior, se rompe la conexión entre éste y el reloj, no pudiendo hacer uso de ninguna de las aplicaciones.

Find Your Phone ha sido desarrollada para ser fácil de usar. Esto se consigue con un menú en modo `gridview`, por el que se navega desplazando el dedo lateralmente. Dispone de cuatro botones, cada uno con una funcionalidad específica y fácilmente reconocible gracias al diseño de éstos.

El objetivo técnico es conocer un poco más sobre la conexión entre el reloj y el teléfono. Esta aplicación utiliza el paso de mensajes como característica principal. A pesar de su nombre, no sólo es una aplicación para encontrar el teléfono, si no, que también sirve para demostrar que es posible manejar los distintos elementos del teléfono como la cámara o la alarma entre otras, a través del reloj.

5.2 Características

Las características que tiene la aplicación y que la diferencian de otras aplicaciones con la misma función de encontrar el teléfono, son:

- Visualización en el reloj de la cámara delantera del teléfono
- Visualización en el reloj de la cámara trasera del teléfono
- Encender el flash en modo linterna desde el reloj
- Hacer vibrar el móvil desde el reloj
- Arrancar la alarma del teléfono sin importar que éste este sin sonido

5.3 Especificaciones

Find Your Phone consta de una pantalla inicial que se corresponde con la previsualización de la cámara del dispositivo móvil, esto es, en la pantalla del reloj se podrá ver lo que esté viendo la cámara del teléfono.

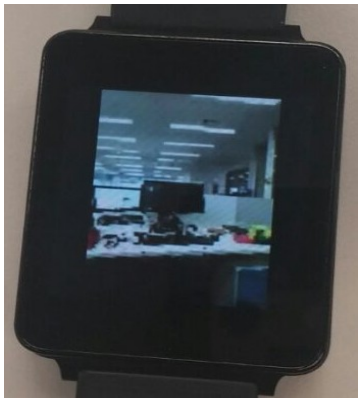

Figura 32. Visualización en el reloj de la cámara del teléfono.

La Figura 32, corresponde con la vista previa de la cámara de un teléfono móvil visualizada en un reloj inteligente, utilizando la aplicación Find Your Phone.

Si desplazamos el dedo hacia la izquierda navegaremos entre las distintas opciones del menú

- Opción 1
    - Dispone de un botón, que si es pulsado se encarga de conmutar entre la cámara frontal y la trasera del teléfono. Para volver a la visualización de la cámara tendríamos que desplazar el dedo hacia la derecha y volveríamos a la pantalla inicial con el cambio de cámara ya realizado.

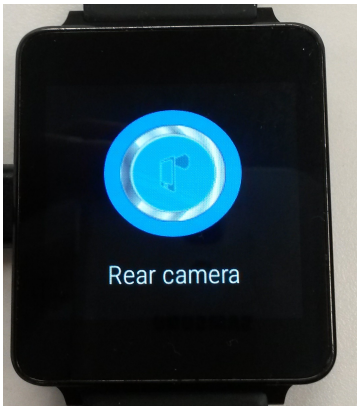

Figura 33. Botón para cambiar entre cámaras

- Opción 2

    - Botón de apagado y encendido del flash en modo linterna. Hay que destacar que si la cámara se encuentra activada en la opción frontal, la opción de flash no surtirá efecto. Una vez activado el flash deberemos desplazar el dedo hacia la derecha hasta la pantalla principal y es ahí cuando el flash de nuestro teléfono se activará.

Figura 34. Botón para encender el flash en modo linterna.

- Opción 3

    - Botón de apagado y encendido de la alarma del teléfono. Si este botón se acciona la alarma se encenderá sin necesidad de volver a la pantalla principal de la aplicación como en el caso de los dos botones anteriores. No importa que el móvil este en silencio, la alarma se activará. Ésta podrá ser desactivada tanto en el teléfono como desde el reloj.

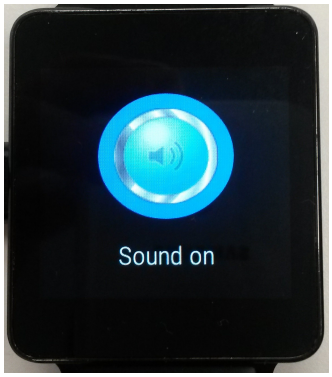

Figura 35. Botón para encender la alarma del teléfono

- Opción 4

    - Botón de apagado y encendido del modo vibración del teléfono. Al pulsarlo el teléfono comenzará a vibrar en series de 10 segundos de duración.
      No podrá ser apagado durante una de las series, justo después de terminar la serie surtirá efecto la acción de apagado.
      Tampoco es necesario desplazarse hacia la pantalla principal para que el móvil comience a vibrar. Es el mismo caso de la alarma.

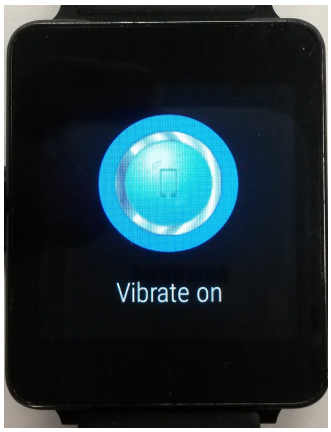

Figura 36. Botón para activar la vibración del teléfono.

5.4 Desarrollo técnico

Find Your Phone es una aplicación cuyo núcleo es el envío de mensajes entre el reloj y el teléfono, las opciones del menú mostradas en las figuras anteriores, al ser pulsadas envían un mensaje al teléfono, y dependiendo del mensaje se realiza una acción específica en el teléfono.

El primer paso es obtener una instancia de un cliente de Google Play Services, para poder tener acceso a las APIs de Google. Para ello, lo primero es importar todas las librerías de Google que vayamos a usar, tanto las que contienen funciones comunes como las específicas para wearables.

```
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.wearable.MessageApi;
import com.google.android.gms.wearable.MessageEvent;
import com.google.android.gms.wearable.Node;
import com.google.android.gms.wearable.NodeApi;
import com.google.android.gms.wearable.Wearable;
```

Bloque de código 8. Importar las principales librerías de Google.

Se añaden al AndroidManifest.xml los permisos para utilizar hardware del teléfono desde el reloj.

```
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.SET_ALARM" />
<uses-feature android:name="android.hardware.camera" />
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-feature android:name="android.hardware.camera.flash" android:required="false"
```

Bloque de código 9. Permisos para utilizar hardware del teléfono en el AndroidManifest.xml

Se especifica el servicio encargado de escuchar los eventos que ocurren dentro del AndroidManifest.

```
<service android:name=".DataLayerListenerService">
```

Bloque de código 10. Clase encargada de escuchar y reaccionar frente a los eventos que ocurren.

En el siguiente bloque de código de 11, en el método `onCreate()` se crea la conexión el cliente de Google Play Services. Como se explicó anteriormente, en el apartado de Google Play Services, es necesario implementar los métodos `addConnectionCallBacks()` y `addConnectionFailedListener()` antes de llamar al `connect()`. Se aprecia que dentro del método `onConnected()` hay una llamada a `findPhoneNode()`, éste se encarga de buscar los nodos conectado con el cliente de Google Play Services.

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(new GoogleApiClient.ConnectionCallbacks() {
                @Override
                public void onConnected(Bundle connectionHint) {
                    Log.d(TAG, "onConnected: " + connectionHint);
                    findPhoneNode();
                    Wearable.MessageApi.addListener
                   (mGoogleApiClient, mMessageListener);
                }
                @Override
                public void onConnectionSuspended(int cause) {
                    Log.d(TAG, "onConnectionSuspended: " + cause);
                }
            })
            .addOnConnectionFailedListener(
             new GoogleApiClient.OnConnectionFailedListener() {
                @Override
                public void onConnectionFailed(ConnectionResult result) {
                    Log.d(TAG, "onConnectionFailed: " + result);
                }
            })
            .addApi(Wearable.API)
            .build();
    mGoogleApiClient.connect();
```

Bloque de código 11. Conectarse a un cliente de Google Play Services.

Una vez que la conexión está establecida, dependiendo de la acción del usuario, tenemos cuatro acciones distintas, por lo que después de ser filtradas, se envían al teléfono. El bloque de código 12, es un ejemplo de la acción de vibrar y la posterior llamada al método que se encarga de enviar el mensaje al teléfono, `sendToPhone()`.

```
  private static int currentVibrate = 0;
     private void doVibrate(int arg0) {
        currentVibrate = arg0;
        sendToPhone("vibrate " + String.valueOf(arg0), null, null);
 }
```

Bloque de código 12. Función que envía al teléfono la acción de vibrar.

La función `sendToPhone()` es la encargada de enviar la información del reloj al teléfono. Está función ya conoce el identificador del nodo con el que tiene que conectarse, por lo que simplemente, con el cliente de Google, el identificador del nodo y la información que se desea mandar, ya es capaz de realizar el envío.

```
private void sendToPhone
 (String path,
 byte[] data,
 final ResultCallback<MessageApi.SendMessageResult> callback) {
    if (mPhoneNode != null) {
        PendingResult<MessageApi.SendMessageResult>
       pending = Wearable.MessageApi.sendMessage
       (mGoogleApiClient, mPhoneNode.getId(), path, data);
        pending.setResultCallback
        (new ResultCallback<MessageApi.SendMessageResult>() {
            @Override
            public void onResult(MessageApi.SendMessageResult result) {
                if (callback != null) {
                    callback.onResult(result);
                }
                if (!result.getStatus().isSuccess()) {
                    if(D) Log.d
                   (TAG, "ERROR: failed to send Message: " + result.getStatus());
                }
            }
        });
    } else {
        if(D) Log.d(TAG, "ERROR: tried to send message before device was found");
    }
}
```

Bloque de código 13. Envío de información del reloj al teléfono.

Después de enviar la información, las acciones de codificación se realizan en el teléfono. Se realiza el proceso inverso. Se dispone de una clase que escucha los eventos que ocurren, es la que recibe el mensaje, lo filtra, y realiza la acción correspondiente. Siguiendo el ejemplo, ahora se ejecutaría la acción de vibrar.

5.5 Arquitectura

En la figura 37, se puede ver en forma de diagrama, las diferentes clases que conforman el módulo *wear*. Como se mencionó anteriormente, cada proyecto Android Wear, dispone de dos módulos, el módulo *mobile* y el módulo *wear.*

Consta de dos clases que hacen referencia a los fragmentos que conforman el menú. `ActionFragment` y `CameraFragment`, en la primera se define el formato común que van a tener los elementos dentro del menú, y en la segunda, se define el elemento que hace referencia a la previsualización de la cámara en el primer elemento del menú. Ambas heredan de la clase `Fragment`.

La clase `MenuAdapter` utiliza los fragmentos anteriormente mencionados, y personaliza cada posición del menú con un elemento diferente, en concreto cinco elementos, uno para la previsualización de la cámara y cuatro para las distintas acciones de la cámara.

La clase `MainActivity`, que hereda de la clase `Activity`, es la encargada ejecutar todos los componentes, es la que da sentido a todo la anterior. Se ocupa de inicializar el menú, de manejar los eventos, y lo más importante, de enviar la información del reloj al teléfono.

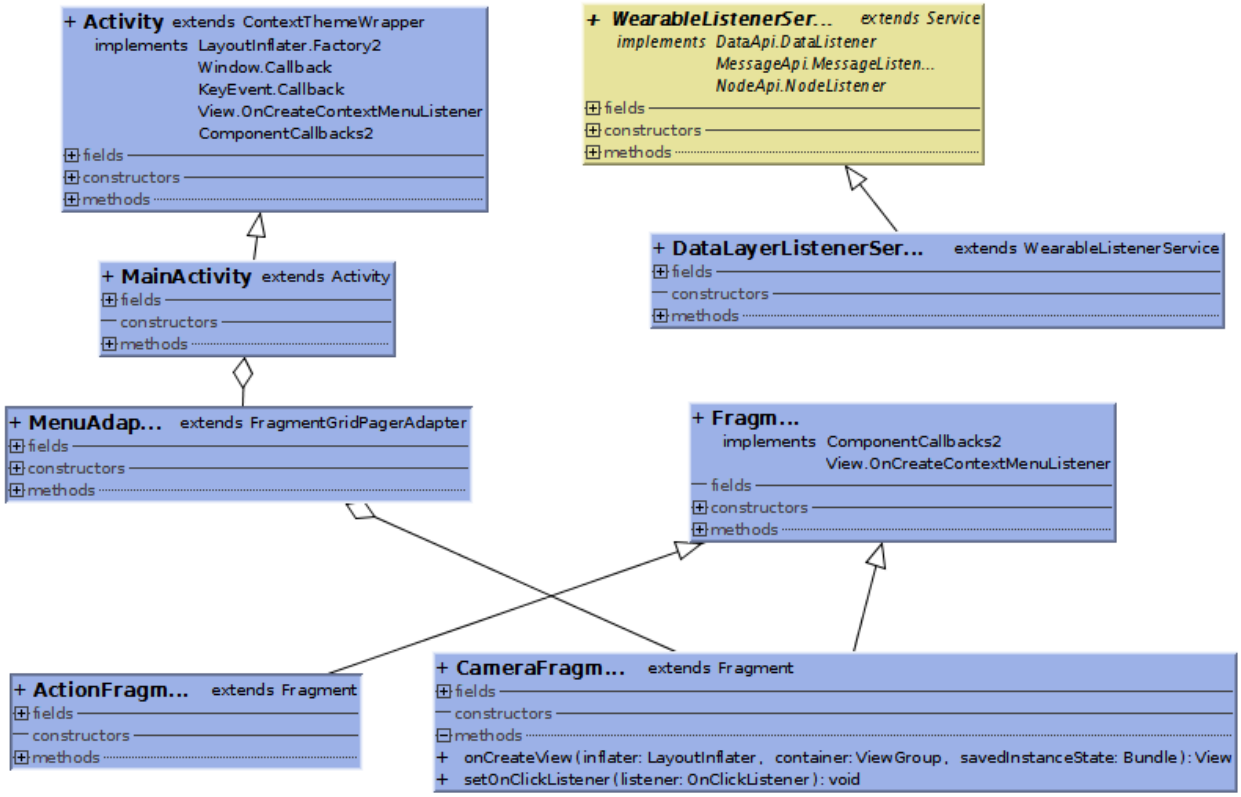

Figura 37. Diagrama de clases del módulo wear.

`DataLayerListenerService` es la clase que se encarga de recibir la información que proviene del teléfono, para ello es necesario que herede de `WearableListenerService`.

El módulo correspondiente al móvile, *mobile*, consta de dos clases, la clase principal encargada de ejecutar la aplicación e inicializar los componentes, `MainActivity`, y la clase que escucha los eventos que ocurren, `DataLayerListenerService`.

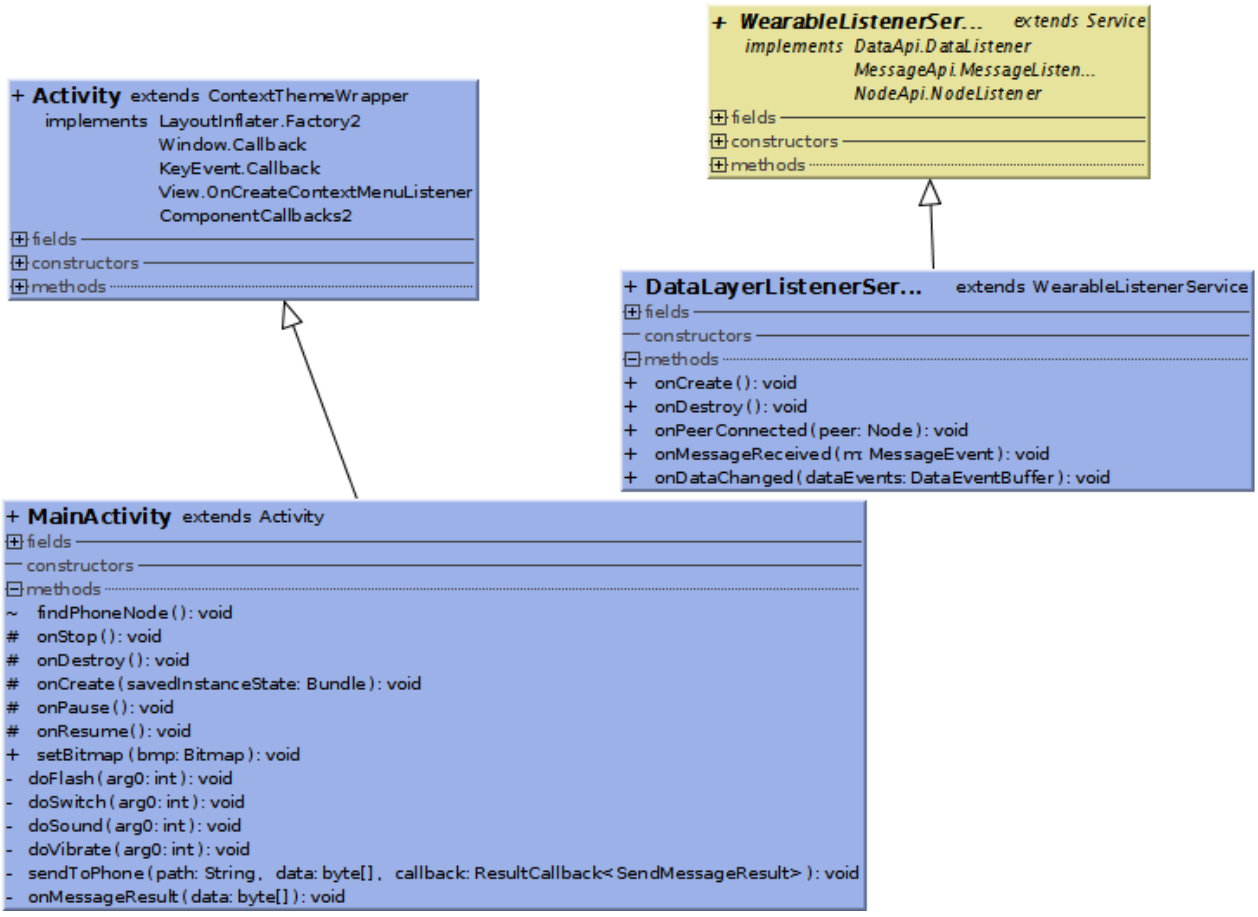

Figura 38. Arquitectura del módulo *mobile*.

La arquitectura del módulo *mobile*, es la misma para las dos aplicaciones.

5.6 Caso de uso

La función principal de esta aplicación es la de facilitar la vida a las personas, acortando el tiempo que dedican a buscar su teléfono.

Con las nuevas características que incluye Find Your Phone comparadas con otras aplicaciones similares que ya hay en el mercado, será más fácil encontrarlo.

Desde tu reloj, podrás visualizar lo que el teléfono esta viendo, de este modo podrás encontrarlo fácilmente. La funcionalidad de poder cambiar la cámara desde el reloj, es porque siempre una de las dos estará inutilizada dependiendo de la posición en la que se encuentre el teléfono.

Si el teléfono se encuentra en un lugar oscuro o con poca luz, la opción del flash será determinante para encontrarlo. Este se quedará en modo linterna, lo cual facilitará su búsqueda.

En caso de que ninguna de las dos opciones anteriores no hayan sido satisfactorias, todavía nos quedarán dos opciones bastante eficaces. Estas son, hacer sonar la alarma o activar el teléfono en modo vibrador. La alarma se encenderá por más que el teléfono se encuentre en modo silencio.

6. Aplicación Social Media Opener

En este capítulo se explicarán las funciones básicas de esta aplicación y se analizará en detalle el desarrollo técnico realizado.

6.1 Alcance y Objetivos

Social Media Opener es una aplicación cuya misión principal es facilitar la apertura de las páginas principales de las redes sociales mediante el uso de la voz.

Una vez abierta la aplicación en el teléfono podrás abrir desde el reloj las redes sociales más usadas sin necesidad de escribir nada o buscar el icono en el teléfono.

6.2 Características

Las características principales de esta aplicación son:

- Conectividad entre el reloj y el teléfono a través de la voz
- Apertura de las redes sociales más usadas del momento.

6.3 Especificaciones

Social Media Opener no consta de una interfaz compleja, pues su función principal es la de mantener conectividad con el teléfono, y enviarle a éste el mensaje recibido mediante la voz.
Con lo cual, tiene un botón con el icono de un micrófono, que al ser pulsado inicia la conexión con el teléfono y se queda a la espera intentado reconocer la voz durante un periodo de 10-15 segundos y si no vuelve a la pantalla principal.

Figura 39. Pantalla de inicio de la aplicación.

Está configurada para abrir las tres redes sociales más usadas actualmente.

- Instagram
- Facebook
- Twitter

El usuario tendrá que pulsar el botón de “Hablar ahora” y posteriormente indicar que aplicación desea abrir mediante la voz, como por ejemplo “Facebook”.

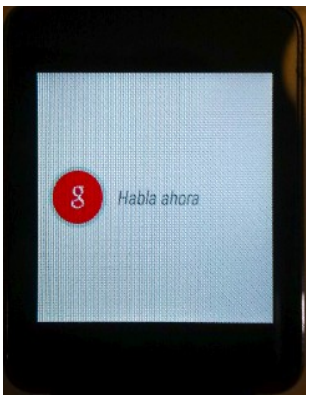

Figura 40. Pantalla que utiliza el reconocimiento de voz de Google.

Figura 41. Reconocer de voz, capturando el mensaje “Facebook”.

El reloj mandará el mensaje al teléfono y éste se encargará de realizar la acción que corresponda. En este caso, será abrir la página principal de Facebook.

Figura 42. Página principal de Facebook.

6.4 Desarrollo técnico

El desarrollo técnico se basa en el manejo de las funciones propias de las funciones de reconocimiento de voz que provee Google y de establecer la conexión entre el reloj y el teléfono que hemos mencionado en el desarrollo técnico de la aplicación Find Your Phone.

La función `checkVoiceRecognition()` es la que se encarga si tenemos la funcionalidad de reconocimiento de voz disponible.

```
public void checkVoiceRecognition() {
       PackageManager pm = getPackageManager();
       List<ResolveInfo> activities = pm.queryIntentActivities(
       new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH), 0);
       if (activities.size() == 0) {
           speak.setEnabled(false);
           showToastMessage("Voice recognizer not present");
       }
   }
```

Bloque de código 14. Función encargada de comprobar el reconocimiento de código.

La función `onActivityResult()` es la encargada de obtener el mensaje de voz en formato texto, y dependiendo del mensaje que sea, realizar la acción correspondiente.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE)
    {
        if (resultCode == RESULT_OK) {
            ArrayList<String> textMatchList =
           data.getStringArrayListExtra(
            RecognizerIntent.EXTRA_RESULTS);

            if (!textMatchList.isEmpty()) {
                String voiceMessage = StringUtils.join(textMatchList.iterator(), "#");
                voiceMsg = voiceMessage;
            }
            if(voiceMsg.equalsIgnoreCase("facebook")){
                openFacebook();
            }
            if(voiceMsg.equalsIgnoreCase("twitter")){
                openTwitter();
            }
            if(voiceMsg.equalsIgnoreCase("instagram")){
                openInstagram();
            }
            Wearable.MessageApi.sendMessage(
      apiClient, remoteNodeId, voiceMsg, voiceMsg.getBytes());
        }

    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Bloque de código 15. Función que obtiene el mensaje de voz y realiza una acción

Se puede observar que realiza una llama a `Wearable.MessageApi.SendMessage()`, que es la función que envía el mensaje al teléfono, y es este el que finalmente filtra el mensaje para abrir la red social indicada por el usuario, como se ve en el bloque de código 15.

La función onMessageReceived del bloque de código 16 se encuentra en el módulo *mobile*, recibe el mensaje que envía el teléfono y realiza una acción determinada.

```
private MessageApi.MessageListener mMessageListener =
 new MessageApi.MessageListener() {
    @Override
    public void onMessageReceived (MessageEvent m){
        if(D) Log.d(TAG, "onMessageReceived: " + m.getPath());
        lastMessageTime = System.currentTimeMillis();
        Scanner s = new Scanner(m.getPath());
        String command = s.next();

        if(command.equalsIgnoreCase("facebook")) {

            openFacebook();
        }
        else if(command.equalsIgnoreCase("instagram")) {

            openInstagram();
        }
        else if(command.equalsIgnoreCase("twitter")) {

            openTwitter();
        }
        else if(command.equals("received")) {
            long arg0 = 0;
            if(s.hasNextLong()) arg0 = s.nextLong();
            displayTimeLag = System.currentTimeMillis() - arg0;
            if(D) Log.d(TAG, String.format("frame lag time: %d ms", displayTimeLag));
        } else if(command.equals("stop")) {
            moveTaskToBack(true);
        }
    }
};
```

Bloque de código 16. Función en el teléfono que recibe el mensaje que envía el reloj y realiza una acción determinada.

6.5 Arquitectura

La arquitectura de Social Media Opener es bastante simple. El módulo *wear* utiliza la clase principal `WearActivity` donde se encuentran todos los métodos relacionados con la captura y procesado de la voz, y los que se encargan de enviar la información al teléfono. También hereda de la clase `Activity`.

La clase `DataLayerListenerService` es completamente análoga a la de la aplicación FindYourPhone.

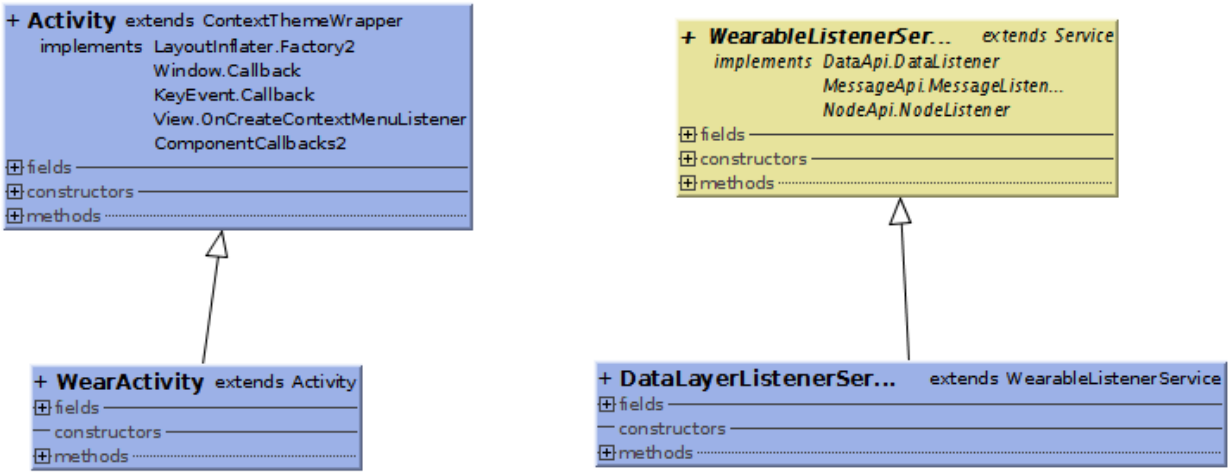

Figura 43. Arquitectura del módulo *wear*.

La arquitectura del módulo *mobile* es completamente análoga a la de la aplicación Find Your Phone.

6.6 Caso de uso

Social Media Opener es una pequeña demostración de cómo se abren muchas de las aplicaciones en dispositivos *wearables*. No todas tienen implementado esta funcionalidad.

La posibilidad de indicar acciones al teléfono a través del reloj mediante el uso de la voz es una forma cómoda de interactuar con los dispositivos, y también ofrece muchas posibilidades para futuras aplicaciones. Esta aplicación es una ligera aproximación a lo que se puede llegar a hacer con esta nueva forma de comunicación reloj-teléfono.

7. Planificación y desarrollo

Este apartado, consta de un análisis de las horas empleadas en el proyecto. Haciendo un desglose por las distintas fases que ha pasado.

7.1 Planificación

En esta sección se detalla el proceso de planificación que ha tenido la realización de este trabajo de fin de grado. Hay que tener en cuenta, que no todos los días se ha empleado el mismo tiempo, por lo que la estimación no es del todo lineal.

La siguiente tabla muestra de una forma organizada el desglose de las actividades con su fecha de inicio y de fin y el número de días aproximados que hicieron falta para cada actividad.

Actividad	**Fecha de inicio**	**Fecha de fin**	**Duración(días)**
Propuesta	19/11/2014	09/12/2014	20
Análisis	11/12/2014	02/02/2015	53
Definición	10/02/2015	09/04/2015	59
Diseño	16/04/2015 02/07/2015	30/04/2015 08/07/2015	20
Implementación técnica	04/05/2015 10/07/2015	25/06/2015 29/07/2015	71
Pruebas	27/05/2015 16/07/2015	30/06/2015 05/08/2015	55
Documentación	09/12/2014 06/08/2015	30/01/2015 29/09/2015	92

Tabla 2. Desglose de las actividades del proyecto.

A continuación se especifica en qué consiste cada actividad:

1. Propuesta

Intercambio de correos electrónicos con la tutora de este proyecto, Celeste Campo, para obtener más información acerca de los objetivos.

2. Análisis

Estudiar la novedosa tecnología Android Wear, con sólo algunos meses de vida. El entorno de desarrollo y estilo de aplicaciones para wearables

3. Definición

Pensar las posibilidades que ofrece Android Wear para explotar sus características principales, y así pensar las aplicaciones para desarrollar.

4. Implementación técnica

Desarrollo de las aplicaciones.

5. Pruebas

Las pruebas no sólo se realizan al finalizar las aplicaciones, si no también se van realizando pruebas a medida que transcurre el desarrollo de éstas.

6. Documentación

Creación de este documento.

7.2 Diagrama de Gantt

La planificación que se ha mencionado anteriormente, en un diagrama de Gantt.

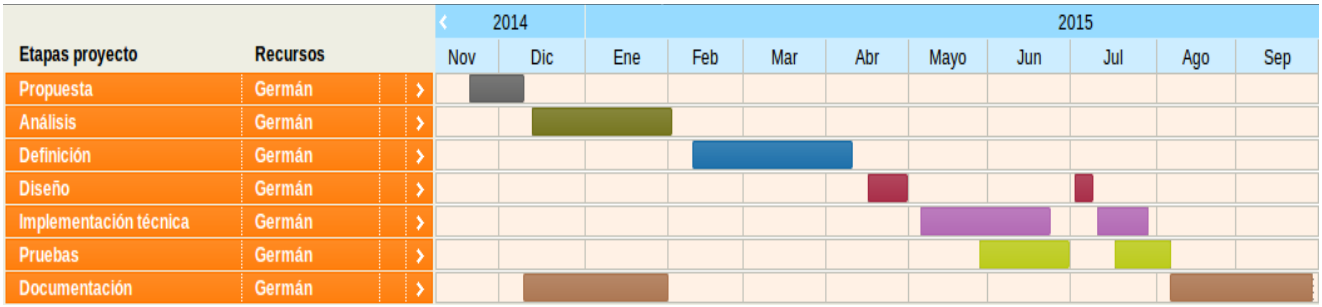

Figura 44. Diagrama de Gantt.

7.3 Presupuesto

Gracias al desglose anterior se puede calcular el número de días totales, que son, 370, a esta cifra hay que restarle 30 días de vacaciones repartidos a lo largo del desarrollo del proyecto y 90 días en concepto de fines de semana. Por lo que el número de días reales es de 250 días, las horas dedicadas a la realización de este proyecto no han sido uniformes debido a tener que compatibilizarlo con asignaturas de la universidad y trabajo a jornada completa en una empresa externa. Se toma como unidad, una jornada laboral de 6 horas, quedando un número total de 1500 horas.

Se necesitan tres perfiles para llevar a cabo este proyecto, analista, programador y diseñador.

En concepto de material se necesita un reloj inteligente, un teléfono móvil y ordenador.

Concepto	Número de días	Número de horas	Coste por hora(€)	Coste total(€)
Analista	53	318	24	7632
Programador	71	426	20	8520
Diseñador	20	120	24	2880
LG G Watch				179
Samsung Galaxy s6				579
HP Pavilion				849

Tabla 3. Desglose de costes del proyecto.

La suma total necesaria para la realización de este proyecto es de 20.639 €.

8. Entorno socio-económico y marco regulador

En este capítulo se va a realizar un estudio sobre la situación económica y el marco regulador que afecta a los dispositivos wearables.

8.1 Entorno socio-económico

Las últimas noticias económicas apuntan en la misma dirección, la recuperación de la economía española. Gracias a la mejora de la actividad económica en España y al crecimiento del consumo de los ciudadanos, el empleo en el sector servicios creció a un ritmo que no se veía desde hace ocho años, crece la producción industrial a su mejor nivel en el último año, los bancos conceden más créditos, por lo que el dinero fluye, el consumo crecerá y empujará la economía[41].

Todas estas noticias son muy esperanzadoras para cualquiera de los mercados emergentes, como el caso de los wearables. Por el contrario, recientes estudios, afirman que los relojes inteligentes no estan teniendo la aceptación deseada. Hay que recordar que es una tecnología muy nueva, y por ahora, la mayoría de las funciones que aportan los relojes inteligentes nos las proporcionan nuestros teléfonos, que siempre llevamos encima. Por ejemplo, el lanzamiento del Apple Watch, causó mucho revuelo, pero ya es oficial que ha pasado de vender 200.000 relojes al día a vender 20.000[42].

El gran fracaso cuando hablamos de wearables son las Google Glass, cuyo proceso se encuentra detenido. Había muchas expectativas con este wearable que no se han cumplido todavía. Hay que tener en cuenta que la forma de esta gafas no es muy aceptada socialmente todavía[43].

El motivo principal, como ya se ha comentado en esta memoria, es que los wearables han llegado al mercado antes de que realmente estemos preparados para utilizar todo su potencial, esto sumado a su elevado precio, hacen que no terminen de conectar con la sociedad.

La función de los werables más aceptada por la sociedad, es su funcionalidad como complemento deportivo. En la figura 45 se pueden ver los distintos usos que la gente da a los wearables.

Figura 45. Usos de los wearables[44].

En la figura 46 podemos ver que la producción de wearables que predomina es la que tiene fines deportivos.

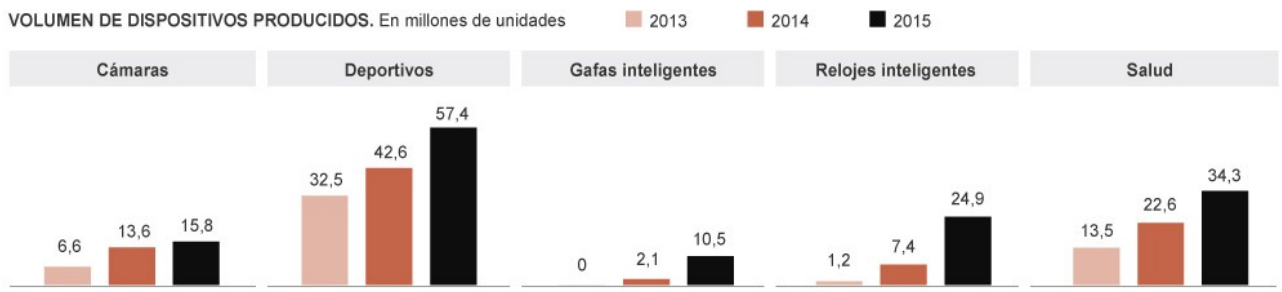

Figura 46. Volumen de dispositivos producidos[44].

8.2 Marco Regulador

Dada la naturaleza de las aplicaciones desarrolladas para este proyecto, están sujetas a la regulación de la Ley Orgánica de Protección de Datos Personales 13/1999 de diciembre, que regula la protección de datos en internet y se encarga de que se cumplan aspectos como los que se encuentran en este documento[45].

La aplicación Find Your Phone hace uso de la cámara del teléfono por lo que tine que ajustarse a la normativa. A su vez, la aplicación Social Media Opener, proporciona acceso a las redes sociales, por lo que también tiene que estar bajo la protección de datos para que no se utilicen los datos personales sin conocimiento de los usuarios.

9. Conclusiones

En este capítulo se encuentra una valoración personal sobre la realización del proyecto, así como las posibles alternativas de diseño que pueden tener o llegar a tener las aplicaciones desarrolladas.

9.1 Valoración personal

Las conclusiones extraídas de las realización de este proyecto son muy positivas. Elegir una tecnología con menos de un año de vida siempre supone un reto. Supone encontrarse con poca información, a veces confusa, supone aprender a base de probar tú mismo, supone hacer una gran labor de investigación para ver realmente qué es la tecnología y qué es lo que se espera de ella.

Un punto importante a destacar, fue el lanzamiento del entorno de desarrollo de Google, Android Studio. Esto supuso dedicar bastante tiempo para aprender a manejar esta nueva herramienta, tiempo en el que se detuvo la investigación sobre Android Wear, para investigar sobre este entorno. Android Studio incluye un emulador, en el cual deposité demasiada confianza, hasta darme cuenta que era inviable realizar una aplicación más compleja que el clásico *hello world*. Una vez descartada la opción del emulador, decidí comprar un reloj inteligente real, lo cual significó un gran avance en el proyecto. La velocidad para hacer las pruebas aumento considerablemente así como todas las opciones que aporta un dispositivo real.

Se han afianzado los conceptos de la programación de aplicaciones Android. Mis conocimientos sobre Android se resumen en la asignatura de Aplicaciones Móviles, por lo que me interesaba mucho seguir aprendiendo sobre una tecnología tan puntera como Android, y conocer Android Wear prácticamente desde sus inicios.

Mi opinión sobre los smartwatches, es que son dispositivos con mucha funcionalidad, pero su precio es elevado para no ser todavía un dispositivo independiente y fundamental como lo puede ser el teléfono. Las nuevas lineas de trabajo de las empresas de este sector están logrando cada vez más acercarse a este objetivo, pero todavía faltan unos años para que esto se cumpla.

9.2 Alternativas de diseño

Las dos aplicaciones tienen margen de mejora, margen para añadir nuevas funcionalidades.

Para la aplicación Find Your Phone

Podría utilizarse para prevenir el robo, y para tener pruebas si éste se produce. Esta aplicación hace uso de la cámara del teléfono, por lo que se podrían añadir dos funcionalidades extra. Un nuevo botón que al ser pulsado haga que la cámara del teléfono empiece a grabar vídeo, y otro que saque fotos. Este material sería guardado en la memoria del reloj, y de esta forma poderlo usar en un posible juicio.

Para la aplicación Social Media Opener

La primera mejora sería incluir más redes sociales como Tuenti, Tumblr, etcétera. La siguiente sería poder indicarle que quieres abrir un perfil en concreto, por ejemplo, diciendo “Facebook Andrea García” y que se abra el perfil indicado. Tener la posibilidad de enviar mensajes diciendo al reloj “Mensaje de Facebook a Andrea García”.

Como se puede ver, las posibilidades para las dos aplicaciones son infinitas, esto es un pequeño ejemplo de lo que los wearables conseguirán en el futuro.

10. Referencias

En este capítulo se encuentran las referencias a la información consultada para la realización de este proyecto.

Referencias principales

[1] http://dazeinfo.com/2014/06/16/samsung-group-smartwatch-sales-2014-galaxy-gear-us/ Preferencias de uso de smartwatches.

[2] https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/ Tendencias de los sistemas operativos para móviles.

[3] https://en.wikipedia.org/wiki/Smartwatch Definición de smartwatch.

[4] https://www.youtube.com/watch?v=MP1gvGcXcLk Demostración Google Glass.

[5] http://appleinsider.com/articles/15/02/25/just-68m-smartwatches-sold-in-2014-at-an-average-price-of-189 Smartwatch group.

[6] http://www.hardwarezone.com.sg/feature-hwmhardwarezonecom-tech-awards-2015-readers-choice-results/readers-choice-awards-consumer-electronics-part-2-3 Mejor marca de smartwatch.

[7] http://seekingalpha.com/article/3019896-intel-inside-the-luxury-tag-heuer-android-smartwatch Ventas de Samsung.

[8] https://www.youtube.com/watch?v=4bxLJkZbLJk Gear S.

[9] https://es.wikipedia.org/wiki/Tizen Tizen.

[10] http://www.macrumors.com/2015/07/31/apple-watch-sales-below-analyst-expectations/ Ventas por debajo de lo esperado.

[11] http://www.apple.com/es/shop/buy-watch/apple-watch-sport/caja-de-38-mm-de-aluminio-en-oro-rosa-y-correa-deportiva-lavanda?product=MLCH2TY/A&step=detail Modelos de Apple Watch.

[12] http://perfectchoiceblog.com/blogmxes/index.php/applewatchenmexico/ Modelo Apple Watch.

[13] https://www.google.es/search?q=moto+360&client=ubuntu&hs=yDI&channel=fs&source=lnms&tbm=isch&sa=X&ved=0CAcQ_AUoAWoVChMIzovS-7uVyAIVhzgaCh1OIAMo&biw=997&bih=516 Modelos Moto 360.

[14] https://www.google.com/landing/now/ Google Now.

[15] https://es.wikipedia.org/wiki/Google_Now#Funcionalidad Google Now.

[16] http://actualidadwatch.com/precios-apple-watch/ Precios Apple Watch.

[17] http://www.elmundo.es/economia/2015/07/13/55a3ed0de2704e81658b4596.html Distintos tipos de wearables.

[18] http://www.xatakamovil.com/sistemas-operativos/de-cupcake-a-marshmallow-asi-han-sido-las-versiones-de-android-a-lo-largo-de-su-historia Historial de versiones, Vídeo Android Marshmallow.

[19] https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android Arquitectura Android.

[20] http://androideity.com/2011/07/04/arquitectura-de-android/ Arquitectura Android.

[21] https://es.wikipedia.org/wiki/Android_Runtime_%28ART%29 ART.

[22] http://www.asociacionaepi.es/android-4-4-kitkat-cambia-su-maquina-virtual-a-art-por-que/ ART.

[23] http://www.edgardandrea.com/archivo-de-manifiesto-en-aplicaciones-android/ Archivo Manifiesto.

[24] http://ocw.uc3m.es/ingenieria-telematica/aplicaciones-moviles/material-de-clase-2/android Transparencias Android.

[25] https://developers.google.com/android/guides/overview Google Play Services.

[26] https://developers.google.com/android/reference/packages APIs Google Play Services.

[27] https://developers.google.com/android/guides/api-client Conectarse a Google play.

[28] https://developers.google.com/android/guides/api-client Código para conectarse a Google Play. Services

[29] https://play.google.com/store/apps/details?id=com.google.android.wearable.app&hl=es Watch Faces.

[30]http://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html Clase Notification.Builder.

[31] https://developer.android.com/training/wearables/notifications/creating.html Notificaciones.

[32] http://www.xatakandroid.com/sistema-operativo/asi-es-android-wear Notificaciones Android.

[33] http://mirrors.sipsik.net/developer.android.com/wear/notifications/remote-input.html Voz en las notificaciones.

[34] https://developer.android.com/training/wearables/apps/voice.html Acciones de voz.

[35] Libro Professional Android Wearables.

[36] http://developer.android.com/tools/studio/index.html#build-system Android Studio.

[37] https://developer.android.com/sdk/index.html#top Descargar Android Studio.

[38] http://developer.android.com/training/wearables/data-layer/index.html Envío y sincronización de información.

[39] http://tools.android.com/tech-docs/new-build-system/user-guide Gradle.

[40] http://www.gottabemobile.com/2014/07/09/how-to-enable-developer-options-on-android-wear/ Reloj en modo desarrollador.

[41] http://www.larazon.es/etiquetas/noticias/meta/la-recuperacion-economica Recuperación económica.

[42] http://www.elmundo.es/economia/2015/07/19/55a7cbbd46163f724b8b4592.html Artículo Apple Watch.

[43] http://www.tecnologia.net/wp-content/uploads/2015/01/Google-Glass.jpg Foto de Google Glass.

[44] http://www.elmundo.es/grafico/economia/2015/03/14/5504ad0ae2704eb8378b4576.html Dispositivos creados.

[45] http://noticias.juridicas.com/base_datos/Admin/lo15-1999.t1.html#a1 Ley de protección de datos.

[46] https://en.wikipedia.org/wiki/Android_Wear Historia de Android Wear

[47]https://developer.android.com/sdk/index.html#top Android Studio

Otras referencias consultadas

https://developer.android.com/training/wearables/apps/index.html Creando aplicaciones para wearables

http://www.bloomberg.com/news/articles/2015-09-03/why-the-smartwatch-hype-machine-is-running-five-years-fast Los relojes se han adelantado 5 años.

http://descargarplay.es/historia-android/ Historia de Android y versiones

http://descargarplay.es/historia-android/ Historial de versiones

https://www.android.com/history/ Historial de versiones

http://androideity.com/2011/07/07/la-maquina-virtual-dalvik/ Máquina Virtual Dalvik

https://es.wikipedia.org/wiki/Dalvik Máquina Virtual Dálvik

https://www.jetbrains.com/ruby/help/inline-debugging.html Depuración en línea